Universidade de Aveiro
Departamento de Eletrónica,
Telecomunicações e Informática
2016

José Luís
da Silva Rosa

Detecção de Ataques de Redirecionamento BGP do Lado dos Clientes

Customer-Side Detection of BGP Routing Attacks

Universidade de Aveiro
2016

Departamento de Eletrónica,
Telecomunicações e Informática

José Luís
da Silva Rosa

Detecção de Ataques de Redirecionamento BGP do Lado dos Clientes

Customer-Side Detection of BGP Routing Attacks

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia de Computadores e Telemática, realizada sob a orientação científica do Doutor Paulo Jorge Salvador Serra Ferreira, Professor auxiliar do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro, e do Doutor António Manuel Duarte Nogueira, Professor auxiliar do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro.

Para a minha família, estando perto ou longe.

o júri / the jury

presidente / president

Prof. Doutor André Ventura da Cruz Marnoto Zúquete
professor auxiliar do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro

vogais / examiners committee

Prof. Doutor Rui Jorge Morais Tomaz Valadas
professor catedrático do Departamento de Engenharia Electrotécnica e de Computadores do Instituto Superior Técnico

Prof. Doutor Paulo Jorge Salvador Serra Ferreira
professor auxiliar do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro

agradecimentos /
acknowledgements

Aos meus pais, a quem palavras não fazem justiça. Eles que sempre apoiaram, acreditaram e sustentaram esta escolha e agora vêem o fruto do seu sacríficio.

À Jessica, por aguentar 5 anos de vida de engenheiro com compreensão, paciência e verdadeiro amor.

Ao João e à Karina, verdadeiros irmãos que nada me negaram ao longo desta caminhada.

À restante família, incluindo os que deixam saudade, mas em especial à minha avó Raquel, sempre admirada pelo exemplo de luta e vontade.

Aos professores Paulo Salvador e António Nogueira, por me incutirem o gosto pelas redes e por me orientarem nesta dissertação.

A todos os meus amigos, colegas e restantes professores que se cruzaram comigo ao longo deste curso e que, de uma maneira ou de outra, influenciaram este percurso e este trabalho.

Palavras Chave BGP, redes, segurança, encaminhamento, deteção

Resumo

A utilização diária da Internet tornou-se uma rotina que foi assimilada pelas pessoas sem considerarem a complexidade interna desta gigante rede. Até um certo ponto, o Border Gateway Protocol é o que mantem toda esta conectividade possível apesar de ser um protocolo defeituoso por natureza.
Em 2008, um ataque Man-In-The-Middle foi pela primeira vez apresentado ao grande público e desde de então mais técnicas para explorar este protocolo e obter tráfego alheio de forma ilícita foram dadas a conhecer. Mesmo que o desvio não aconteça com natureza maliciosa, mas sim devido a um erro de configuração, este é um problema que deverá ser enfrentado.
Alguns provedores de serviço e institutos de investigação já apresentaram propostas para novos protocolos e/ou sistemas de monitorização, mas estes estão atrasados no seu desenvolvimento ou apenas afetam a camada superior da rede, deixando utilizadores e um grande número de empresas que estão ligadas a um provedor sem meios para agir e sem informação sobre o encaminhamento do seu tráfego.
Nesta dissertação, é apresentado, concebido e implementado um sistema que atinge uma monitorização ativa do BGP através da medição do tempo médio de viagem de vários pacotes enviados de várias localizações, através de uma rede mundial de sondas, e do processamento dos resultados obtidos, permitindo que todos os interessados possam ser alertados.

Keywords BGP, networking, security, routing, detection

Abstract

The daily use of the Internet has become a routine that many people absorbed into their lives without even thinking about the insides of this gigantic network. To an extent, the Border Gateway Protocol is what is keeping all this connectivity together despite being a very flawed protocol due to its design.
In 2008 a Man-In-The-Middle attack was first presented to the general audience and ever since more techniques were reported to use the protocol to obtain traffic illicitly. Even if the routing deviation does not occur via a malicious intention but due to some poorly configured router, this is a problem that must be tackled.
Some network providers and research institutes already presented some drafts for new protocols or monitoring systems but they are late into deployment or only affect the top layer of the network, leaving users and most part of the companies connected to the provider impotent and without any proper information about the routing of their traffic.
In this dissertation a system is presented, implemented and deployed, achieving an active monitorization of BGP through measurements of the average travel time of several packets sent to various locations by a worldwide set of Probes and the collected results processed allowing all concerned actors to be alerted.

Contents

List of Figures

List of Tables

Acronyms

AH Authentication Header

ARP Address Resolution Protocol

AS Autonomous System

ASN Autonomous System Number

BGP Border Gateway Protocol

BR Border Router

CIA Confidentiality, Integrity and Availability

DDoS Distributed Denial-of-Service

DNS Domain Name System

DRF Django REST Framework

DTL Django Template Language

EGP Exterior Gateway Protocol

ESP Encapsulating Security Payload

GTSM Generalised TTL Security Mechanism

GUI Graphical User Interface

HTTP Hypertext Transfer Protocol

ICMP Internet Control Message Protocol

IGP Interior Gateway Protocol

IP Internet Protocol

IRR Internet Routing Registry

ISP Internet Service Provider

MITM Man-In-The-Middle

NLRI Network Layer Reachability Information

ORM Object-relational Mapper

OS Operating System

OSI Open Systems Interconnection

PKI Public Key Infrastructure

REST Representational State Transfer

ROA Route Origination Authorization

RPKI Resource Public Key Infrastructure

RTT Round Trip Time

S-BGP Secure BGP

SCP Secure Copy

soBGP Secure Origin BGP

SQL Structured Query Language

SSH Secure Shell

TCP Transmission Control Protocol

TLS Transport Layer Security

URL Uniform Resource Locator

VPS Virtual Private Server

CHAPTER 1

INTRODUCTION

1.1 MOTIVATION

The Border Gateway Protocol is the protocol in place to connect the Internet in modern days. However, it is a protocol flawed by design in terms of security, which leaves users helpless in case of a detour of their traffic. These defects presented by the protocol affect the confidentiality of data, availability of the communications and integrity of service, usually known as the Confidentiality, Integrity and Availability (CIA) triad of security[1]. Despite existing some solutions and systems that can be employed by network providers and some major corporations, everyday users and companies are exposed to the flaws that inherently exist.

Even if the user is able to detect that his traffic is being rerouted, he can not act on a network level leaving his options reduced to warning his provider and use a higher level of security policies such as the termination of incoming and outgoing data transfers and the usage of encryption in public services that are not using it already.

1.2 OBJECTIVES

This dissertation is part of the DARTEG (Deteção de Ataques de Redirecionamento de Tráfego a uma Escala Global) project[2] and it was developed in the Aveiro site of Instituto de Telecomunicações in the research area of the Advanced Telecommunications and Networks Group (ATNoG)[3]. Alongside to this dissertation, student Mário Pina was also developing his Masters' dissertation in the same field with connections to this work. During the various chapters, his name and work will be referenced when necessary.

The ambition of this work is to implement and deploy a worldwide monitoring system that is able to detect Border Gateway Protocol (BGP) traffic redirections based only on the Round Trip Time (RTT) for each monitored network. The monitorization should be based on a set of Probes deployed in several locations and the measurements should be reported to a central location who acts as a Server and also hosts a website that allows connection by authorised users.

The processing of the results obtained by the various probes should be sensible enough to correctly correlate the past measurements to the incoming result and detect consistent consecutive deviations from the estimated average.

Concerning the Probes operation, they do not require much capabilities or resources, which allow their deployment to be done in Virtual Private Servers (VPSs) spread through various datacenters, which is vital to guarantee a worldwide coverage and a correct detection of routing attacks, since they can only be seen by probes sufficiently apart from the monitored network and/or the attack point.

1.3 STRUCTURE

This first chapter gives an introductory explanation to this dissertation and it is complemented by chapter 2 where the BGP is explained, including its features and flaws, how they can be exploited and what can be done to prevent it. Some detection and alarming services are also explored in that chapter.

In chapter 3 the implemented system is discussed, starting with a general view of the architecture and the components of the system, before discussing the datamodel and proceeding to explain the implementation of those concepts for every component.

Lastly, chapter 4 looks at the Probes used to obtain Results in this dissertation, as well as the Targets that were monitored, before looking at some Results to try and draw relevant conclusions.

Chapter 5 wraps the whole dissertation and gives a look at what is ahead of it.

CHAPTER 2

State of the Art

2.1 the Border Gateway Protocol

The BGP is the standard of the current Internet routing and is used as the *de facto* protocol[4] for connecting the Internet. It was built using all the experience gained during the development and usage of the External Gateway Protocol (the protocol EGP should not be confused with the family of protocols with the same name) and came into development to replace it.

Despite being projected to work as an Exterior Gateway Protocol (EGP), BGP can also work as an Interior Gateway Protocol (IGP), as shown in figure 2.1 (adapted from [5]), but the internal aspect of the protocol will not be covered in this work because it is not prone to the hijacking attacks that are discussed later. From this point onward, whenever BGP is mentioned it should be assumed that it refers to the External part of the protocol.

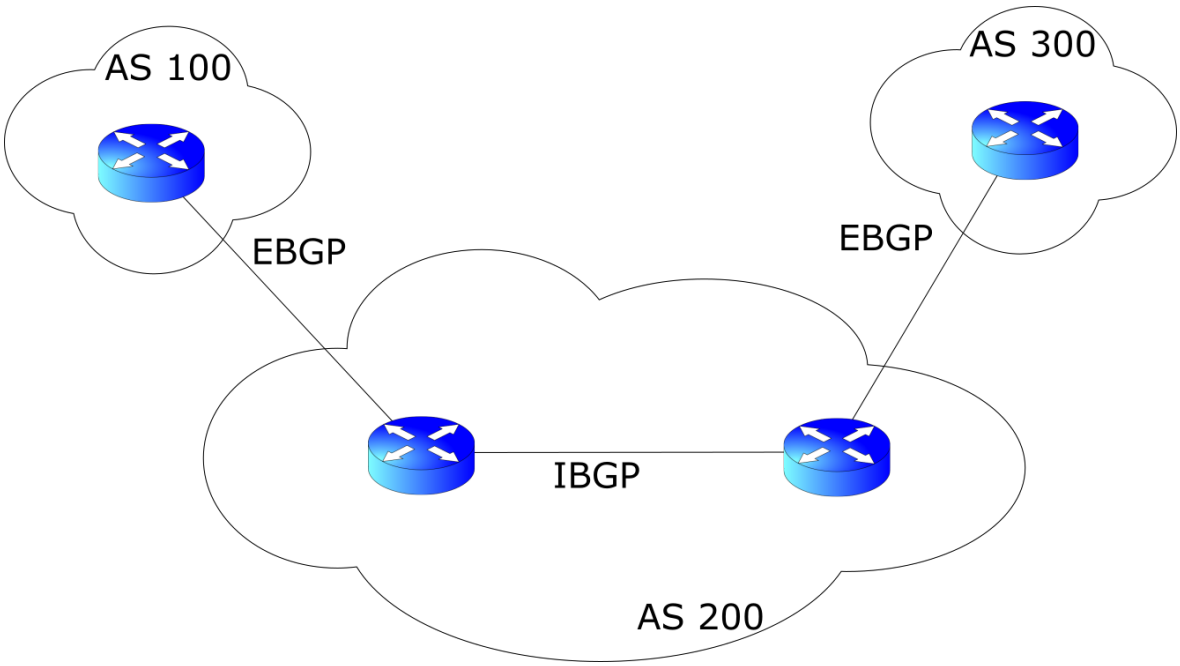


Figure 2.1: External and Internal BGP

The version 4 of BGP lists the primary function of BGP as a mechanism of information exchange between several BGP-speaking systems[6], also known as Autonomous Systems, which Hawkinson, J. et al defined as *"a connected group of one or more IP prefixes run by one or more network operators which has a SINGLE and CLEARLY DEFINED routing policy"* in RFC 1930[7]. These Autonomous System (AS) are identified by an Autonomous System Number (ASN) that can be any number between 1 and 64511 for public ASN and 64512 and 65355 for private ones. Because this is a finite range of ASNs a later proposal was published[8] to extend from 2 bytes to 4 bytes ASN under the format <higher 2 bytes in decimal>[dot]<lower 2 bytes in decimal> but for the purpose of this dissertation the 2 bytes format is used.

BGP operates on the Internet by constructing the reachability table and the network graph according to the information contained in the announcements from other BGP-speaking routers (also called Border Router (BR)) containing the routes to reach others AS, leaving to the receiving router the decision on each route to take, based on some rules as the most-specific IP prefix or the length (in number of hops) of the route. These announcements are made in the UPDATE messages and contain a set of attributes, where the mandatory ones are:

Origin - How the router learned the route. 0 for Internal, 1 for External and 2 for incomplete or unknown origin.

Next Hop - The Internet Protocol (IP) address that should be used for reach the announced AS. If BGP is working as an IGP this attribute should be propagated and not modified.

AS Path - List of all AS that passed the announcement and form the path to reach the original ASes. Whenever a router receives an announcement and redistributes it to his peers it prepends his ASN to the path already present in the message.

An important aspect that will be explored later is that, in order to avoid loops in the announcement transmission, whenever a router sees an route announcement with his own ASN in the message, the router just discards the packet.

Any two BR that form a connection are called, in a general way, *neighbours* but the connections could be of three types[9][10]:

Peer-Peer - Mostly on the core of the Internet, these neighbours pass traffic without expecting monetary compensation but do not announce each other routes to transit or other peers.

Customer-Provider - The customer pays the service provider to be able to announce his routes and transit traffic to the rest of the Internet.

Sibling-Sibling - When two AS are administrated by the same company or authority they can pass traffic without limitations.

These definitions are an important concept to define the *valley-free property* of the BGP routing, which represents the idea that a route that passes through a Provider-Customer or a Peer-Peer edge should not transverse another Peer-Peer edge or Customer-Provider edge. This is a relevant notion for the later discussion of BGP hijacking. Figure 2.2 (adapted from [11]) illustrates a route that breaks this property and is, in fact, the original Proof-of-Concept done by Pilosov and Kapela (In section 2.2 this attack is studied with more detail).

As said before, the relevant information for the creation of BGP paths are sent in the UPDATE messages. These messages contain multiple fields that can be used to indicate routes that are no longer

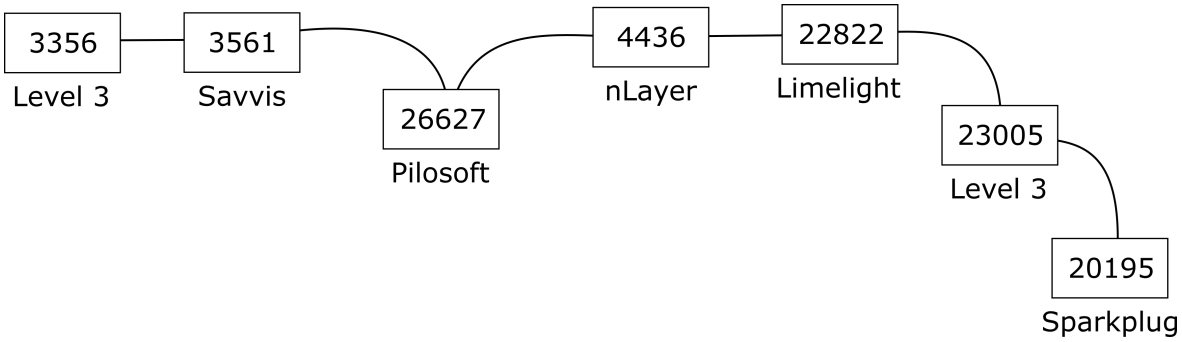


Figure 2.2: Route breaking the "valley-free" property of BGP

acceptable or new routes in the network. The new routes are sent in a field called Network Layer Reachability Information (NLRI). When a new route is perceived by a router, it should add its own ASN to the AS Path before redistributing the route. Alongside with the new routes, a router can also announce several attributes that are not the well-known mandatory ones presented before. One of the attributes that is important to look at, is the Communities attribute and this is classified as an Optional Transitive attribute, that may not be supported by all BGP implementations. However, when it is, it could be used for BGP hijacking, as seen later in the text.

Communities can be simply described as a set of routes who share some property and are grouped to simplify the routing policy application[12]. This attribute is designed to work as a 32-bit value, but some of the values are advised to be treated as reserved. Also, there are 3 values that are global to every AS:

NO_EXPORT (0xFFFFFF01) - This route or group of routes should not be carried to outsiders of a BGP confederation (an AS sub-divided into internal ASs for administration purposes[13]).

NO_ADVERTISE (0xFFFFFF02) - This route or group of routes should not be carried to any BGP peers.

NO_EXPORT_SUBCONFED (0xFFFFFF03) - This route or group of routes should not be carried to any BGP peers outside of the AS even if they are part of the BGP confederation.

Later in this chapter, a form of attack on BGP will use the aforementioned global communities.

2.2 BGP vulnerabilities

With the major aspects of the protocol presented, it is an obvious understanding that BGP was not implemented with security concerns in mind. This created some long-lasting vulnerabilities that are not solvable, only monitarable[14]. All BGP transactions are based on trust[15] and in recent years there was an increase in BGP hijackings caused by either erroneous configurations or malicious intents[16]. These problems impose as a major threat because there is not a great level of awareness to them and the high quantity of UPDATE messages in circulation (in 2009 Renesys[14] reported more than 45.000 UPDATES per minute). With no mechanism for authenticating these messages, anyone with a minimum expertise can forge BGP announcements and insert new routes into the network or produce false AS paths.

However, not all of the redirections that happen on the Internet are illegitimate. One of the challenges of BGP hijacking detection is the detection between perfectly normal redirections, like traffic engineering or physical causes (hardware failures[17], natural causes[18]) and abnormal ones (malicious intents or erroneous configurations[19]). Nonetheless, this kind of BGP attacks are not the first ones to occur and certainly will not be the last ones. Attackers, hackers or simply networking professionals already use techniques such as Distributed Denial-of-Service (DDoS), where the intruder tries to disrupt the access from the user to the system either by overloading the network connectivity or the server's resources[20], or Domain Name System (DNS) blocks (such as the "Great Firewall of China") where the Internet Service Providers (ISPs) or governmental agencies can inject false registries to conduct users to the wrong location[21], thus blocking the access to legitimate sites (for example, blocking Twitter or YouTube for censorship purposes).

When talking about BGP attacks, hijacking is a general way to refer three types of attacks:

Blackholing - Similar to DNS blocks, attackers can receive the traffic intended for the victim and drop it (to Null0 interface, for example), cutting the flow of information. [4] [22]

Impersonating - When the attackers pose as the victim, either by answering to the hijacked traffic or using the announced IP prefixes to send spam e-mails. [4]

Interception - Our main concern, the attackers receives the traffic and can read it (and even modify it) before sending it to the original destination[22]. Interception attacks are generally invisible to the victim and were complex to perform due to the fact that is necessary to have a clean path to the victim during the attack. In 2008, Pilosov and Kapela presented their Man-In-The-Middle (MITM) attack [9] in which a new way to do this 'clean path' was introduced.

Interception attacks work by capturing the traffic intended to the victim, just like in Blackholing or Impersonating, but directing it to the intended receiver after reading and/or modifying it. In 2014, Renesys reported knowing at least three methods to perform this[23], which include Pilosov and Kapela's Man-In-The-Middle, an approach which uses the Communities attribute of BGP and, finally, a technique that explores the announcements only to peers and not to providers in order to take advantage of the blockage to free transit. Starting with MITM, the first step is to convince the remaining ASes that our announcement is the correct one and, most importantly, the best route to the announced IP prefix. This can be accomplished by winning the race conditions that happen at BGP tables, such as the most specific prefix (i.e., a /16 is more specific than a /8, and so the routers should choose it) or the shorter route in terms of size of AS Path. There are more attributes that can influence this choice (Cisco has a Best Path Algorithm [24] with 13 conditions and Juniper has 15 condition [25], for example) but this are the most used ones when someone tries to *win* ("Winning" is defined as gaining the best route to legit announcer) the possession of routes. [9]

The problem exists when trying to do this poisoning of other BGP routers without taking in consideration the delivery of traffic to the original destination. An untreated announcement can lead to every packet that is sent out to be returned in a short period of time because everyone else thinks that the route to reach the attacked AS is that one. Pilosov and Kapela use the *no loop* property of AS Paths to keep a clean path to the victim, analysing the traceroute result and noting which ASN should be prepended to the AS Path. This will cause that when the path that should be clear receives the UPDATE message with the route being hijacked, it will drop that packet because his own ASN is already in it. The final step in the process is to create a static route to the first AS in the "fake" AS Path, redirecting the traffic to the original destination. In figures 2.3 to 2.7 (adapted from [14]), is

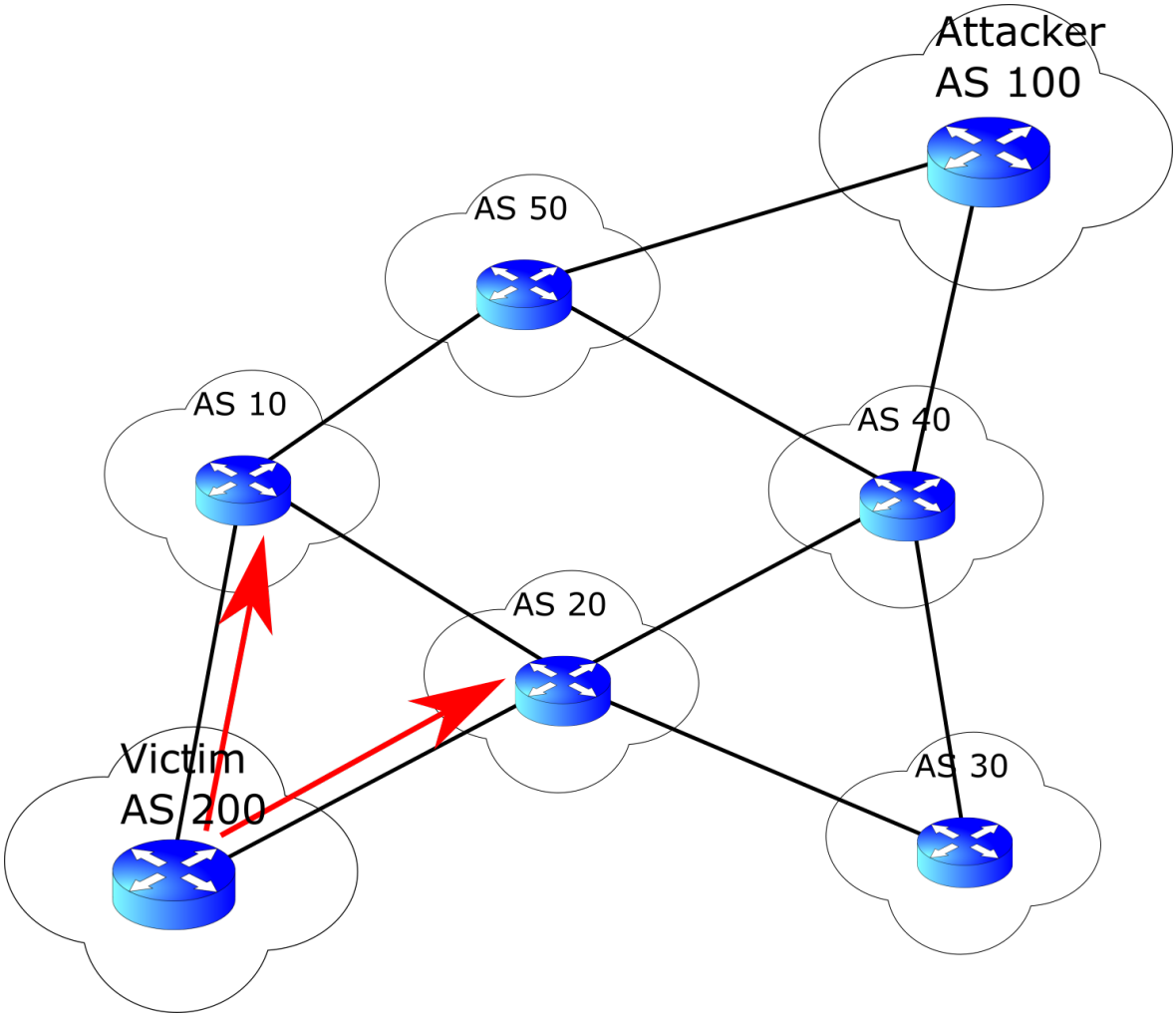


Figure 2.3: Pilosov and Kapela's MITM: Legit announcement

described a hypothetical Man-In-The-Middle attack: AS 200, our victim, starts by announcing himself to his neighbours (figure 2.3), who propagate the announcement to the whole network (figure 2.4) before everyone converges and the best path for each router is chosen (figure 2.5).

At this point, the attacker must analyse the traceroute to the intended target so that he can prepare the malicious announcement with the ASN of the ASes in the path prepend to the AS Path, making the routers in that path discard the packets (figure 2.6). Finally, after the false announcement is propagated, almost everyone in the network is sending the packets, intended to the victim, to the attacker who then redirects it to the victim using the clear path that was preserved (figure 2.7).

Despite all the preparation done before launching the attack, it is relatively easy to detect the hijacking with a simple traceroute that would expose the addresses that the traffic crossed until returning to the original destination. Hijackers can perform a TTL adjusting, which will erase the "new" routers from the traceroute and, with some success, mask the presence of the attacker. This can be achieved by adding enough values to TTL so that the traceroutes do not show the unwanted steps between the sender and the victim [9].

The method, however, does not impact one important aspect of the traffic, which is the time factor. Because BGP uses the Transmission Control Protocol (TCP) sessions to communicate, it is possible to determine the time between transmission using the ACK messages, thus, showing a large jump in

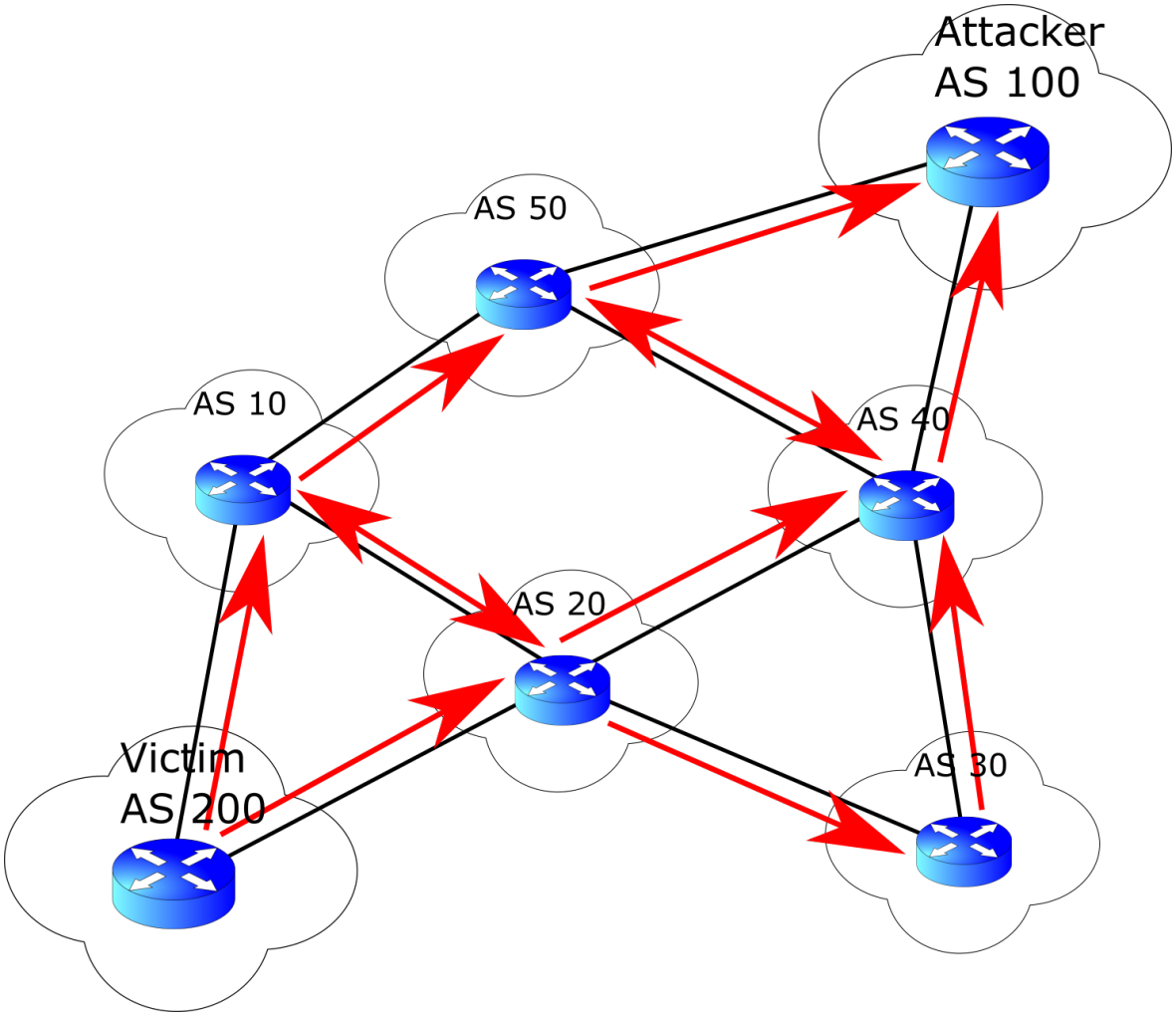


Figure 2.4: Pilosov and Kapela's MITM: Announcement propagation

times when the TTL masking is used and the reroute is big enough to be noticed. In figure 2.8 [9] we can see the complete path that was done by the traffic in the original attack demonstrated, by Pilosov and Kapela, where all the traffic that was destined to DEFCON was rerouted to Pilosoft servers in Los Angeles. When TTL masking was applied, as shown in figure 2.9 [9], 10 steps were erased from the records, but the time is still the same, making it an easy task to a person or system who looks into the results to detect a jump between 28 and 88 msecs [14]. In 2014, Renesys believed that this type of technique was still to be used outside of a testing or proof-of-concept utilisation [23].

Another technique that is reported is the use of BGP Communities to restrict the announcements to upstream providers, making the hijack confined to a certain spatial limit and consists in a riskier approach, since it is needed to fully understand Communities and the way they aggregate and treat announcements. The way this is done is to announce the desired routes using the P:71990 community, which is of the type P:7DNNA [23] [26] being disclosed by WHOIS databases to be a type of community that prevents prefix propagation, where D indicates which peers are blocked in the propagation (1 for international peers and 2 for in-country peers), NN specifies the upstreams (99 stands for all providers while the others numbers represent individual providers) and A defines the action and only takes the value 0 (Do not announce prefix) [23] [26].

The last procedure that was reported [23] involves doing the announcements only to peers and

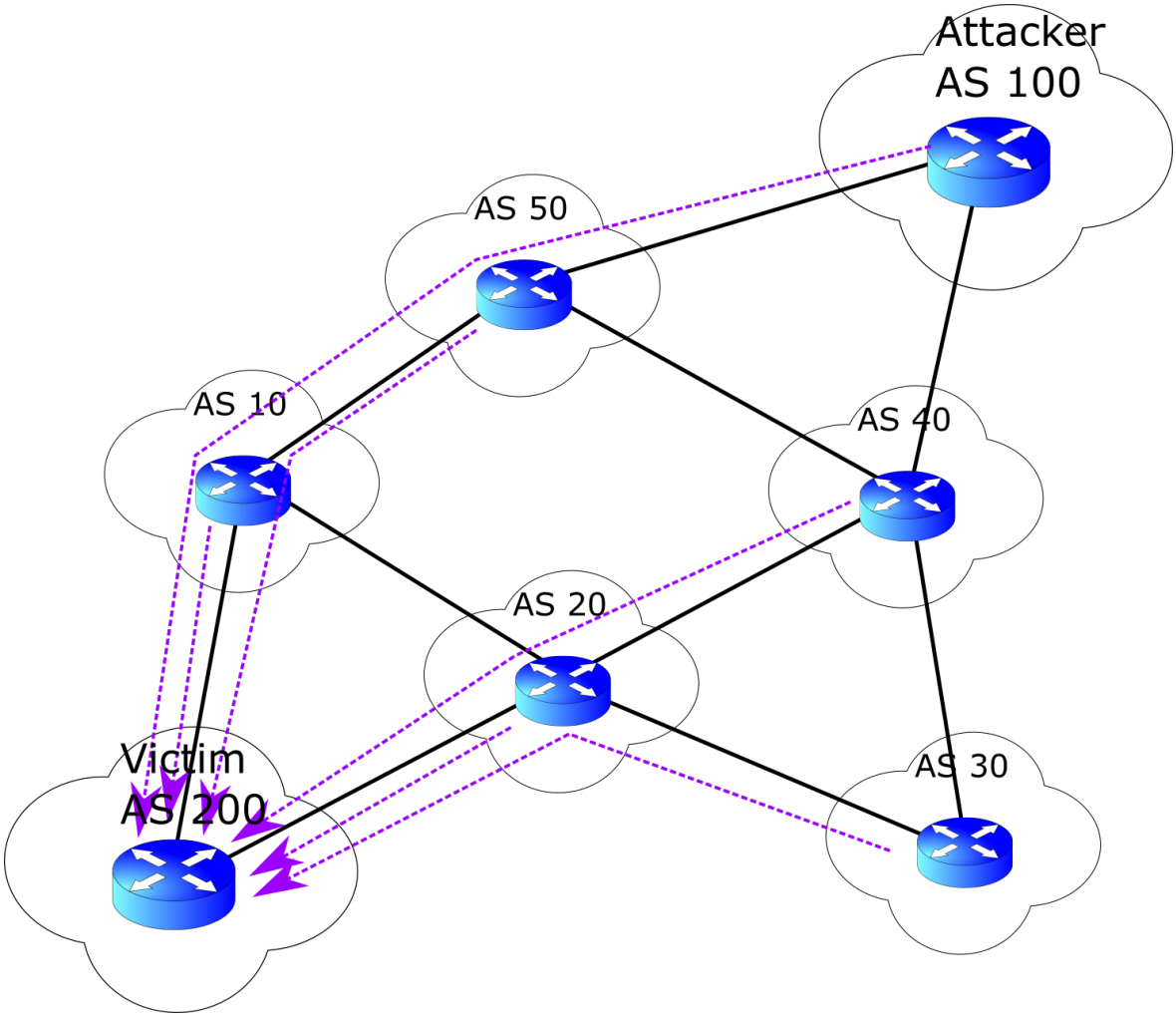


Figure 2.5: Pilosov and Kapela's MITM: Network convergence

not to clients, providers or siblings. Since the peers, per definition, do not transit traffic or redirect announcements, this will have the same effect that the Communities approach and limit the hijacking to nearby Autonomous Systems [23] [26].

Nonetheless, having the knowledge to perform these attacks on the BGP network does not automatically endangers the protocol or its speakers, forcing the attackers to gain access to the network, either directly or remotely. Direct access can be gained by someone who works near the physical routers or someone who infiltrates the locations where the routers are located, while remote access depends (almost always) on some kind of malicious attack that allows the control of the feed. One of the most popular mechanisms to do this, is the Trojan Horse, a piece of software that can dissimulate himself as a legit program to the user and performs malicious actions without his knowledge. [27] In recent year, this Trojans became more specific to deal with the new ways that people browse the Internet, promoting the appearance of "socially engineered Trojans". Apart from Trojans, attackers can also use bugs or failures in the software used in the computers, phishing attacks perpetrated by email or some form of malicious code like viruses, worms and malware [28].

The above mentioned mechanisms and attacks are all in a theoretical perspective. According to Dyn Research (previously Renesys), 2013 was the year that marked the debut of these attacks on the Internet, with more than 150 cities containing at least one victim of attacks [29]. Since

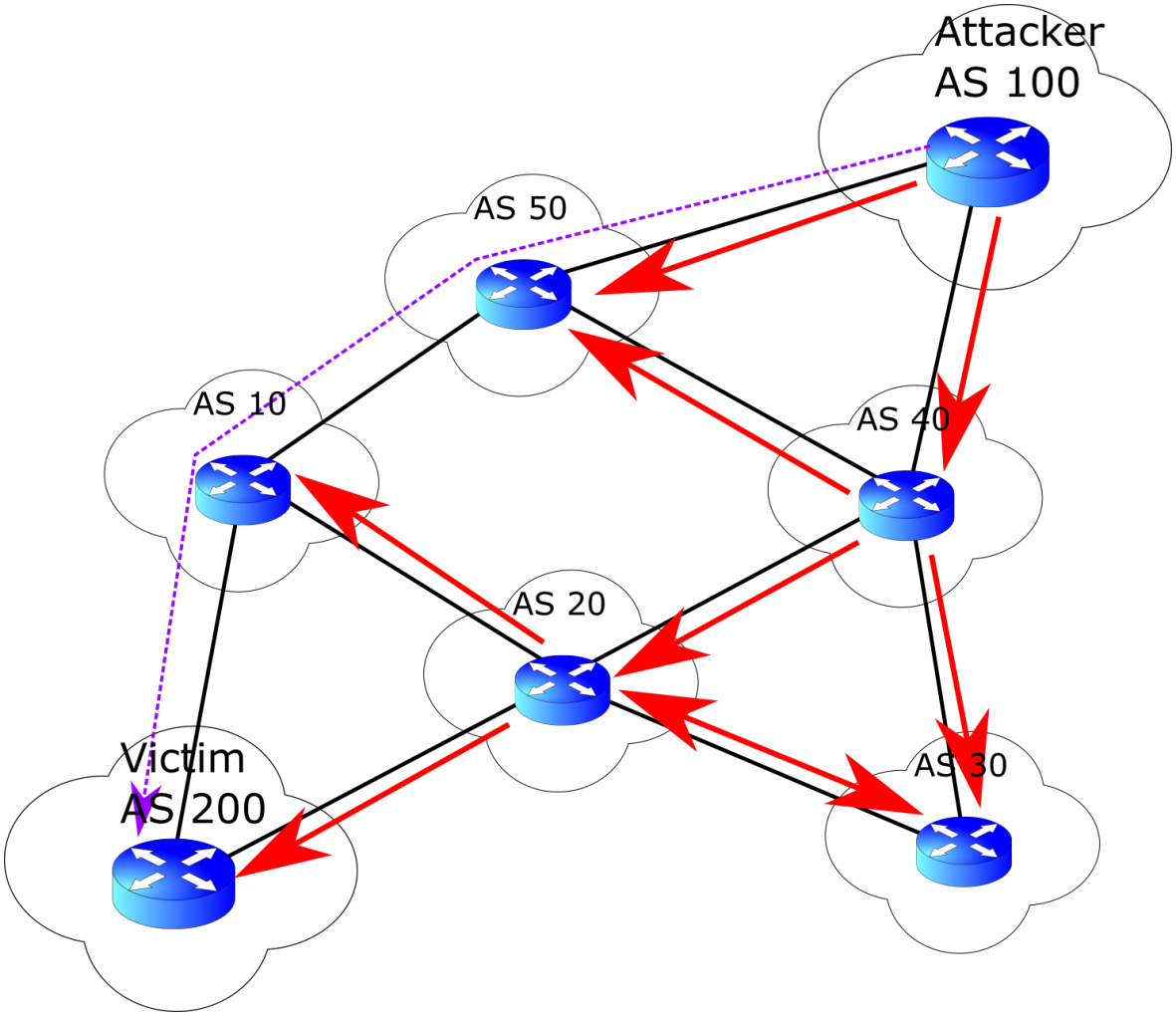


Figure 2.6: Pilosov and Kapela's MITM: False announcement

then several hijacks, intentional or not, have been detected nearly every month, in several locations, including Portugal. One of the most serious cases happened in March 2013 when Vega (an Ukrainian telecommunications provider) started the hijack on prefixes owned by the United Kingdom Atomic Weapons Establishment and the traffic that should directly link Houston, Texas, United States, to the United Kingdom started to take a detour to Frankfurt, Germany, and Kiev, Ukraine, before being returned to the original destination. The hijack lasted close to 90 minutes and it hijacked more than 150 prefixes, where the AWE ones were included. Apart from those, Vega was announcing erroneous prefixes over the course of 5 days and included companies like Walmart, Pepsi Co., the Royal Mail, The Football Association, banks, telecommunications companies, and others[30]. Other incidents were already reported by the same company (Dyn Research), some of which include international traffic like Guadalajara, Mexico to Washington D.C., United States being relayed to Moscow, Russia and Minsk, Belarus before returning to America, or Chicago, Illinois, United States to Tehran, Iran doing a complicated path before arriving at the original destination (figure 2.10). There are, however, reported cases of hijacking happening in traffic flowing in the same city. In 2013, traffic that should transverse two locations in Denver, Colorado, United States, took a huge deviation and choose a path that crossed London, United Kingdom and Reykjavik, Iceland. This is one of the episodes in a larger hijack detected between July 31 and August 19, 2013, that originated in Iceland in 17 distinct times. [29]

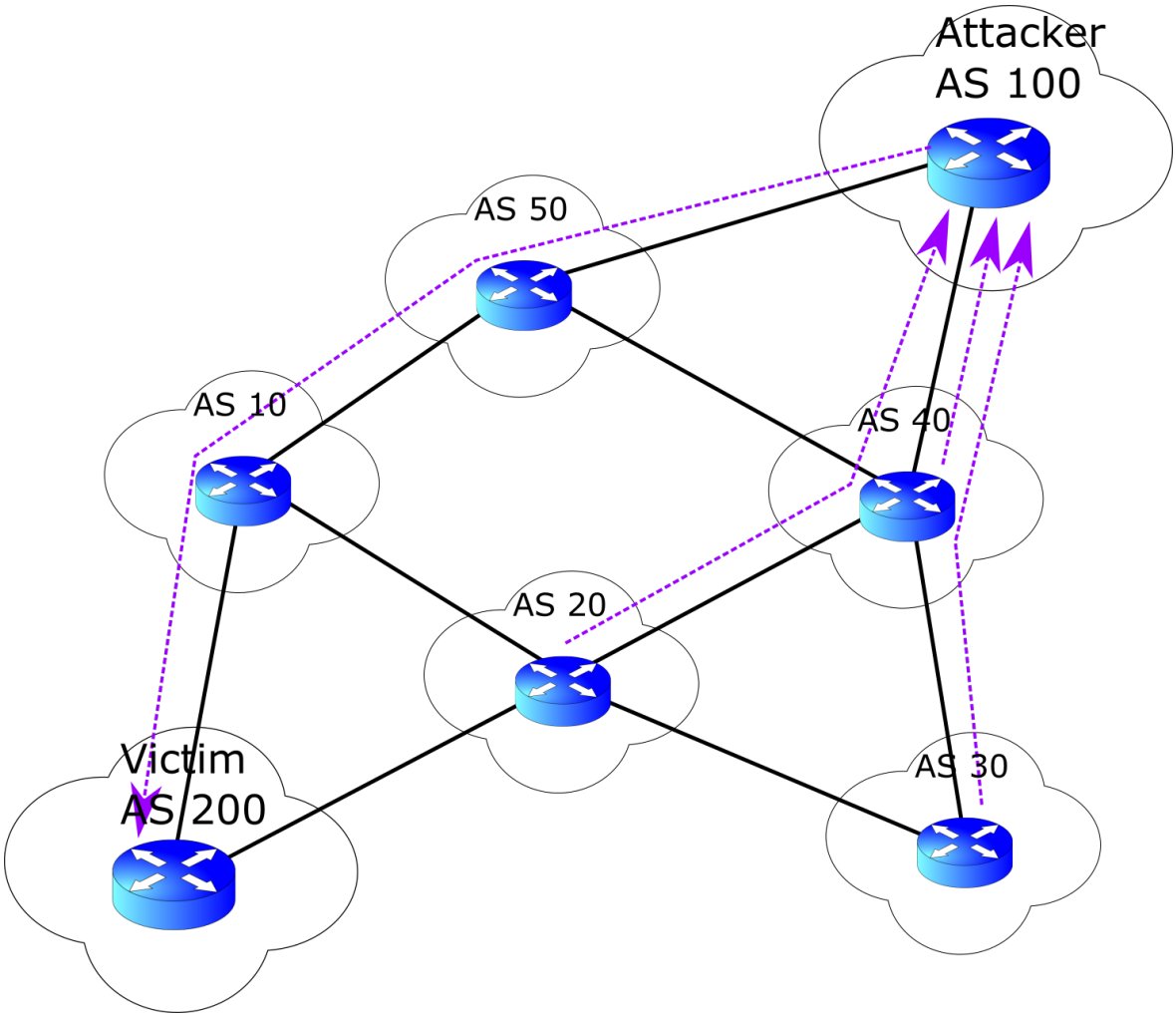


Figure 2.7: Pilosov and Kapela's MITM: Hijacking in progress

```
 2 12.87.94.9 [AS 7018] 4 msec 4 msec 8 msec
 3 tbr1.cgcil.ip.att.net (12.122.99.38) [AS 7018] 4 msec 8 msec 4 msec
 4 ggr2.cgcil.ip.att.net (12.123.6.29) [AS 7018] 8 msec 4 msec 8 msec
 5 192.205.35.42 [AS 7018] 4 msec 8 msec 4 msec
 6 cr2-loopback.chd.savvis.net (208.172.2.71) [AS 3561] 24 msec 16 msec 28 msec
 7 cr2-pos-0-0-5-0.NewYork.savvis.net (204.70.192.110) [AS 3561] 28 msec 28 msec 28 msec
 8 204.70.196.70 [AS 3561] 28 msec 32 msec 32 msec
 9 208.175.194.10 [AS 3561] 28 msec 32 msec 32 msec
10 colo-69-31-40-107.pilosoft.com (69.31.40.107) [AS 26627] 32 msec 28 msec 28 msec
11 tge2-3-103.ar1.nyc3.us.nlayer.net (69.31.95.97) [AS 4436] 32 msec 32 msec 32 msec
12 * * *   (missing from trace, 198.32.160.134 - exchange point)
13 tge1-2.fr4.ord.llnw.net (69.28.171.193) [AS 22822] 32 msec 32 msec 40 msec
14 ve6.fr3.ord.llnw.net (69.28.172.41) [AS 22822] 36 msec 32 msec 40 msec
15 tge1-3.fr4.sjc.llnw.net (69.28.171.66) [AS 22822] 84 msec 84 msec 84 msec
16 ve5.fr3.sjc.llnw.net (69.28.171.209) [AS 22822] 96 msec 96 msec 80 msec
17 tge1-1.fr4.lax.llnw.net (69.28.171.117) [AS 22822] 88 msec 92 msec 92 msec
18 tge2-4.fr3.las.llnw.net (69.28.172.85) [AS 22822] 96 msec 96 msec 100 msec
19 switch.ge3-1.fr3.las.llnw.net (208.111.176.2) [AS 22822] 84 msec 88 msec 88 msec
20 gig5-1.esw03.las.switchcommgroup.com (66.209.64.186) [AS 23005] 84 msec 88 msec 88 msec
21 66.209.64.85 [AS 23005] 88 msec 88 msec 88 msec
22 gig0-2.esw07.las.switchcommgroup.com (66.209.64.178) [AS 23005] 88 msec 88 msec 88 msec
23 acs-wireless.demarc.switchcommgroup.com (66.209.64.70) [AS 23005] 88 msec 84 msec 84 msec
```

Figure 2.8: BGP hijacking without TTL masking

```
 2 12.87.94.9 [AS 7018] 8 msec 8 msec 4 msec
 3 tbr1.cgcil.ip.att.net (12.122.99.38) [AS 7018] 4 msec 8 msec 8 msec
 4 ggr2.cgcil.ip.att.net (12.123.6.29) [AS 7018] 4 msec 8 msec 4 msec
 5 192.205.35.42 [AS 7018] 8 msec 4 msec 8 msec
 6 cr2-loopback.chd.savvis.net (208.172.2.71) [AS 3561] 16 msec 12 msec *
 7 cr2-pos-0-0-5-0.NewYork.savvis.net (204.70.192.110) [AS 3561] 28 msec 32 msec 32 msec
 8 204.70.196.70 [AS 3561] 28 msec 32 msec 32 msec
 9 208.175.194.10 [AS 3561] 32 msec 32 msec 32 msec
10 gig5-1.esw03.las.switchcommgroup.com (66.209.64.186) [AS 23005] 88 msec 88 msec 84 msec
11 66.209.64.85 [AS 23005] 88 msec 88 msec 88 msec
12 gig0-2.esw07.las.switchcommgroup.com (66.209.64.178) [AS 23005] 84 msec 84 msec 88 msec
13 acs-wireless.demarc.switchcommgroup.com (66.209.64.70) [AS 23005] 88 msec 88 msec 88 msec
```

Figure 2.9: BGP hijacking with TTL masking

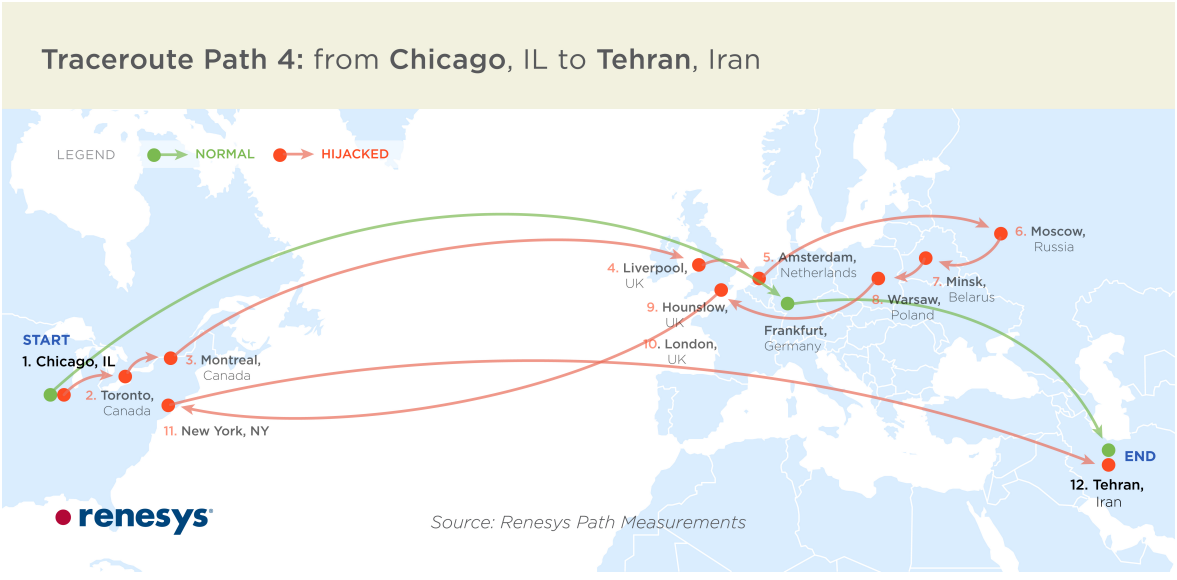


Figure 2.10: Example of an hijacked route

2.3 SECURING BGP

What can be done to prevent the hijacks? Some solutions were already proposed and partially implemented, ranging from updated protocols (BGPSEC, SBGP, So-BGP) to cryptographic signatures on the packets (RPKI) or simple methods that can be implemented by the providers in their own infrastructure, like prefix filtering. All of the mentioned mechanisms can be considered to be active solutions but there are also some methods of passive monitoring that aim to monitor the Internet and BGP announcements and detect erroneous UPDATES messages.

The most radical of solutions would be changing the protocol to a new or updated one that could support verified announcements or cryptographically signed packets. Some tentatives were already made, in the likes of, but not limited to:

Secure BGP (S-BGP) - Development started in 1997 [31], aims to be a three-way method, that applies the use of a Public Key Infrastructure (PKI), a new BGP transitive attribute and IPsec to authenticate and encrypt the packets. The PKI consists of various mechanism, policies and procedures that support two main components, the Certification Authorities and the Public Key Cryptography, in order to ensure a secure communication for people or organisations over the Internet. [32] This infrastructure, applied to S-BGP, authenticates the IP blocks being announced, as well as the AS owner and the BGP routes in the UPDATE message. The new attribute is used to contain the digital signatures to attest the validity of the announcement, together with the PKI information. Finally, IPsec assures the correct encryption for the packets that transverse the Internet, the data integrity and the authentication between BGP routers. [33] [34]

Secure Origin BGP (soBGP) - Another proposal was made by engineers working at Cisco and it was baptised as the Secure Origin BGP. It is based on the notion of digital certificates, making use of three types of certificates to ensure the secure communication between BGP speakers and it adds a new type of message to BGP called SECURITY message. This message will transmit all information that regards the certificates and creates backwards compatibility, since it does not modify any existing message type. Using this messages, the soBGP speakers will trade certificates called Entity Certificates (to ensure the identity of the speaker), Authorization Certificates (to assign IP prefix blocks to the routers) and Policy Certificates (to pass policies for the prefixes between routers on the network). [35] While S-BGP is a more complete and robust solution, soBGP is a much lighter one, existing a trade-off between security and computation time that should be analysed by actors concerned with those protocols. [34]

BGPsec - The most recent, BGPsec, bases some ideas on S-BGP, including the use of cryptographic keys to ensure the correct transmission of routes through BGP speakers. Using a Resource Public Key Infrastructure (RPKI), BGPsec proposes a way to secure route propagation where each speaker signs his messages, creating a set of nested signed messages when the announcements are spread through the Internet. The RPKI also does the association between ASN and IP blocks, causing a chain of trust in conjunction with the signed messages. After having his blocks in the RPKI, an AS can associate a Route Origination Authorization (ROA), i.e. a special signed block issued by the RPKI to the specific AS, to an announcement. The ROAs, the signed nested messages and the RPKI create a simple yet robust method to secure the AS Path and provide some security to the protocol. [36] [37]

These solutions were designed to be applied on the ISP-controlled side of the network, but while they are in a development process there are other measures that can be applied. One example is prefix filtering or route filtering, where AS administrators can enforce rules that prevent their routers to accept announcements that contain erroneous or suspicious information. The basic rule is to filter the prefixes that the AS owns, but even that one is sometimes ignored. [9] Apart from that one, ASes can start filtering using databases like the Internet Routing Registry (IRR), that will cover some, but not all, cases, which provide some degree of associations between ASN and IP blocks. However, these databases are publicly accessible and anyone can insert data into them, making them not 100 percent trustworthy. Even if the administrators do not filter the inbound traffic, they can start filtering the outbound traffic, preventing the spread of erroneous configuration on their side. If everyone filters their traffic the number of cases of unintentional hijack will drop, but, citing Pilosov and Kapela "Until everyone filters everyone perfectly, this door is still open". [9]

Still on the ISP side of actions, there are some actions that can be taken but would generally imply collaboration from two or more peers or an excessive amount of manual work. For example, the use of TCP MD5 Authentication would allow two routers to exchange a BGP (since BGP sessions are supported by TCP) message with an associated payload which has been digested by the MD5 algorithm. The receiving router can digest the content of the message using the same technique and, in the end, compare the result with the digest already in the message. If the results differ, the packet should be dropped because it cannot be trusted. This, however, requires every set of peers to have a manually configured set of keys to execute the digest and that those keys are changed periodically to minimise the risk of attacks. This creates a whole lot of work for the network administrators, which creates a barrier to be adopted by everyone. [38] Another solution, already explored when the BGPsec was addressed, is the use of IPsec, either by using the Authentication Header (AH) or the Encapsulating Security Payload (ESP), to ensure a secure communication between neighbouring routers. Nonetheless, this only tackles the local problems, not preventing route hijacking. Finally, Generalised TTL Security Mechanism (GTSM) can intervene in the case of TTL adjusting (talked before, when the MITM by Pilosov and Kapela was discussed) is used. Since BGP sessions use (generally) TTL = 1, because their neighbours are, in most cases, in a distance of one hop, it is easy to modify this value to hide the tracks of the attack. GTSM uses TTL = 255 (the maximum value for 8 bits) and when a router sees a packet with TTL under 254, it drops it. But, as before, this does not prevent route hijacking on a large scale, being only a small leverage in a large problem. [38]

Apart from the solutions that involve using the Network and/or the Transport layer of the Open Systems Interconnection (OSI) model, some approaches have been presented over the years and frequently they are some form of monitoring systems for the BGP announcements. For this type of solution, it is important to distinguish between data-plane and control-plane solutions, based on the information that they use to formulate the need or not to alert the owner of the AS. The former is characterised by controlling the network, taking decisions based on previous policies, even if the data is not flowing opposed to the latter which consists in directing the traffic and the packets when they truly arrive at the router. Applied to the mentioned solutions, it can exist mechanisms that monitor the Internet based on the policies in place or the data that is flowing on the BGP itself or even both. Using only control plane information provides a solution that can be deployed with ease but can be imprecise when analysing the network because of the lack of BGP data. Using only the data plane also provides a fairly easy system to set up, but is limited by the number and position of the monitoring probes. Lastly, a method that combines the two planes can perform a joint, real-time detection for hijacks but is also limited (in the perspective of the accuracy) to the number and position of the probes.

Table 2.1 (adapted from [39]) synthesises these points [39].

Requirement	Control plane	Data plane	Control and Data plane
Real-time	depending on the sources	✓	✓
Accurate	X	limited by vantage points	limited by vantage points
Light-weight	✓	✓	✓
Easily deployable	✓	X	✓

Table 2.1: Different types of BGP monitoring systems

Examples of this type of solutions are the PHAS [40] or the iSpy. [39] Nonetheless, the three types described before have two major flaws: there is no real incentive to ISPs to deploy the systems, i.e., if very few ASes are running the system there is no real monitoring in place, and they do not have a notification system embedded by default, making the network administrators performing one more task in their routine.

Another proposal is a system capable of performing what was called by the authors as "Promoting and purging routes". In simpler terms, it takes advantage of the mesh of ASes to elect one or more lifesaver routes that would send the correct routes to the infected neighbours. These routes are identified using the RIPE MyASN service and after detecting an attack the lifesavers purge their own BGP table and promote the routes on their neighbours taking advantage of the AS_SET attribute where, by definition, no matter how many ASes are in the AS-Path, its size is always one, making the promoted route have the minimum possible size. [41]

All these approaches share one idea, that the system has access to BGP data, either the inbound-/outbound traffic or the network topology. But, what happen if a simple company signs a contract to an ISP? There is no real information available to these types of situations, apart from that who depends on the TCP rather than BGP itself (round trip time, time to live, traceroute). These companies rely on some services that are offered by third-parties and range from the most basic, free, ones to some services that are commercial level, with costs associated. Some examples of these services are the Internet Alert Registry (discontinued) [42], RIPE NCC Routing Information Service [43], BGPmon [44], Dyn's Internet Intelligence [45] or Watchmy.NET [46]. They may contrast in some aspects, for example, cost (free or paid), support (existence of a SLA or no assistance), response times (in a matter of seconds or hours), data sources (how many and how disperse are the probes), accuracy (the rate of false positives and/or false negatives), among others. [14]

CHAPTER 3

System Architecture and Implementation

3.1 Proposed solution and architecture

In this dissertation, it is proposed a solution based on a Server, built on a modular architecture, and a variable number of Probes. In the Server, different modules will process the data and communicate with each other in order to accomplish the desired monitorization of the BGP, in conjunction with a variable number of Probes, spread worldwide, that will run Scripts uploaded via the website and perform the necessary tests.

Approaching the solution with more detail, figure 3.1 shows the components that are already deployed, how they connect with each other and the protocols or languages used to achieve this.

Starting with the User Module, it aims to store all the information about the users, validate their credentials and permissions in order to give access to parts of the system, and communicate these permissions to other components.

The website and the Graphical User Interface (GUI) use the functionalities provided by the module mentioned before to allow users who are correctly logged in and have the necessary permissions to view, edit, insert or delete data from the system.

Linked to the website but also used by other components, a Storage component allows the Scripts that will be uploaded to be saved on the Server and a Logging component will store all the logs generated by the different components in the course of their functions.

One of the central pieces of the whole system and that is linked to practically all other components is the Database. The database will store all the information regarding Probes, Scripts, Results, Alarms and any information generated by a conjunction of any of these.

The Probe Management component is responsible for all matters related to the Probes, either by creating a new instance in the system, uploading Scripts to the Probes, checking if a Probe is responsive, among other tasks. This component will interact with the Probes via Secure Shell (SSH) and Secure Copy (SCP) (over the SSH channel) in order to provide a secure way of communication between them.

After the Probes are configured and the Scripts running on them, it is necessary to have endpoints on the Server to assure the correct reception of data. For that purpose, two components are in charge of

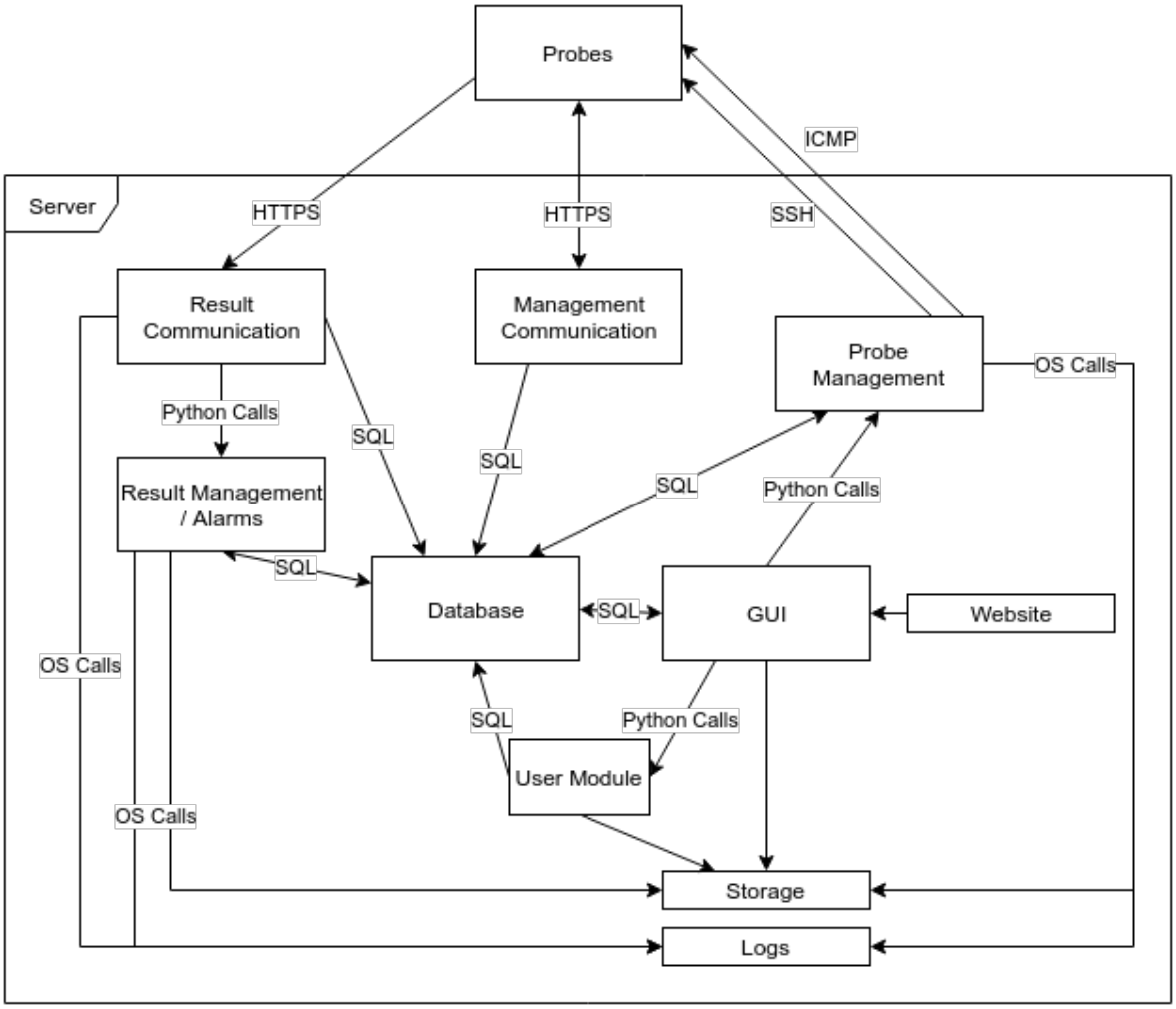


Figure 3.1: Diagram of the system's components

the communication with the Probes. The first one to be addressed is the Management Communication, responsible for receiving data that concerns the utilisation of the Probes and its resources. This component receives signals from Scripts that are monitoring the Probes and can make changes to the data present in the Database, if necessary.

The second component that acts as an endpoint for Probe communication is the Result Communication, responsible for gathering the data taken by the Probes during the execution of the tests and store in the database. After this data is stored, the Result Communication must be able to alert the Result Management component that a new Result is ready to be analysed.

The aforementioned Result Management component, who also aggregates the Alarming component, is in charge of processing the incoming Results, via Result Communication component, and based on the data contained in them, setting up the relevant Alarms, either for one specific Target or for the Probe who collected the Result.

All of these components are on the Server side of the system, but they need a set of Probes in order to have meaningful information that allows their activity to be done. These Probes can be of a variable number but it is important that they are geographically distant from each other to guarantee a worldwide coverage. This is valuable, in the context of the BGP routing attacks, because Probes too close to the attack can be blind to its occurrence. In figure 3.2 it is given a possible configuration of

a set of Probes and it shows the occurrence of a cluster of Probes, this is, two Probes in the same location. When this happens, one of them can be set as a Backup and, in the case of failure by the Active Probe, the Management Communication component can switch their roles in order to maintain a monitorization in that region.

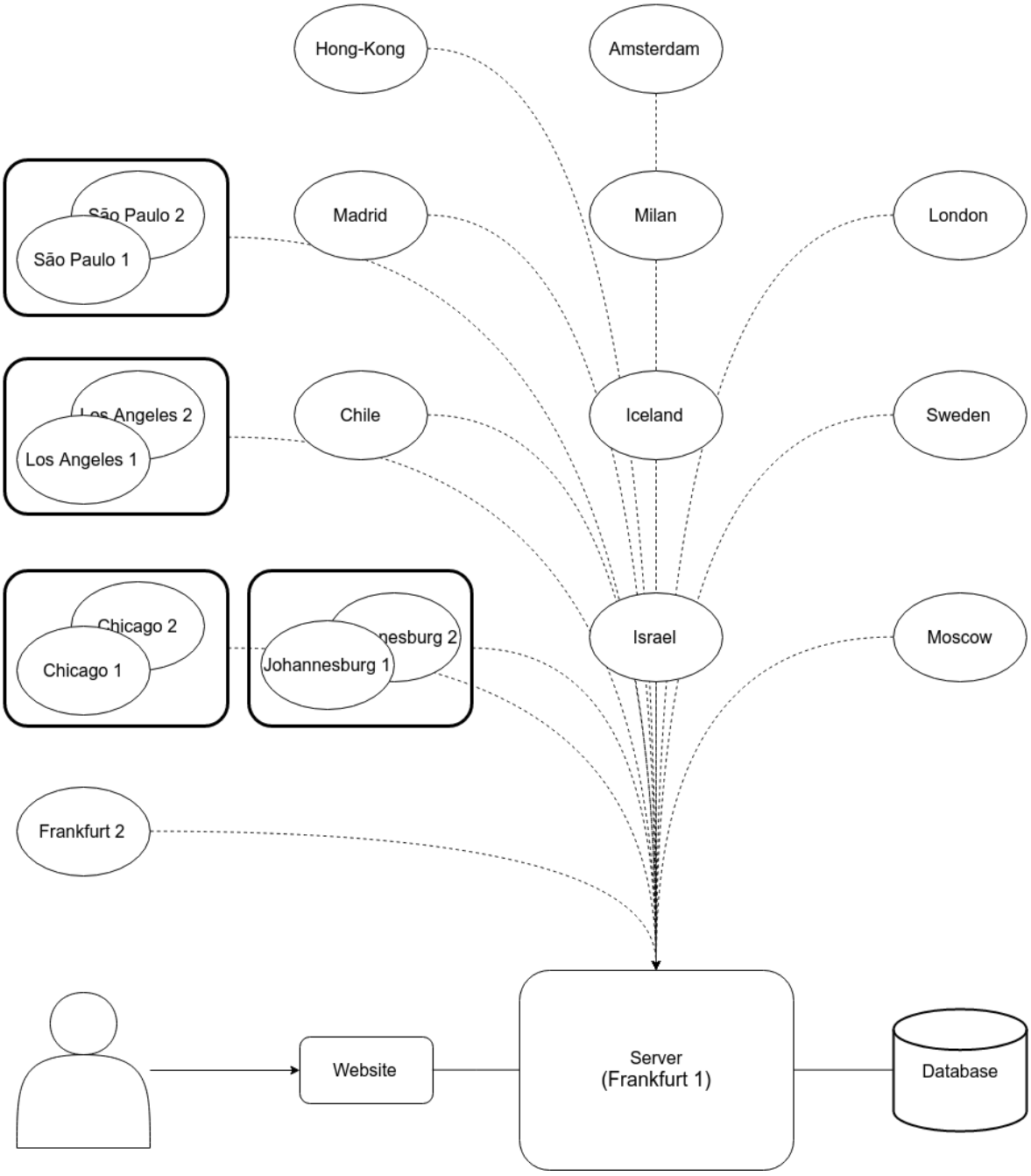


Figure 3.2: Diagram of the Probes connected to the Central Server

3.2 DATA MODEL

To store all the data necessary for the project, a datamodel was designed in order to accomodate all the needs for every component and over the course of the project, the tables and columns of each table were iterated to reflect the state of the development. The diagram shown in Figure 3.3 acts as a Relational diagram, showing the tables and relations between them. In this diagram, it is already possible to observe implementation-specific concepts, such as Django's tables and, as such, the datatypes presented were generated by Django to better accommodate the Model definition in MariaDB, as it will be discussed ahead in section 3.3.

It is worth noting, that all tables have a numeric integer ID that serves as the primary key for that table. For this point onward, this field is implicit when talking about a table.

Starting with the tables generated by default at the start of a Django project, and that were used in this project (more about that in subsection 3.3.1), there are three tables worthy of mention. The Group table stores the name of each Group of users. A Group is a simple way of giving permission to a set of Users, instead of having to do it one by one. Linked to the Group, the table for User Groups does the association between a Group ID and a User ID and is the table that stores the information of what Users are in which Group. Finally, the User table stores all the data relevant to the User in the system like the password of the User, date of his last login, first and last name, username and email that are provided at the register and the date were the User joined the system. It also contains three verification fields to check if the User is considered a Superuser (subsection 3.3.1), Staff or if the User is Active.

The User table is a fundamental piece in this model, establishing ownerships in the different tables across the model. One of these tables, is the Probe table, and stores all the information directly regarding the Probes linked to the system, such as, a text identifier, the IPv4 and IPv6 addresses, the status of the Probe (Active or Backup), the User that owns the Probe and a field that states if the Probe ever encountered an anomaly. Connected to the Probe table, the Backup table stores two Probe IDs and creates a relationship of the type Active-Backup between two Probes.

Alongside the Probes, two more tables save information in regard to the various types of Scripts that can be uploaded via the website, one for Processing Scripts and another one for Monitoring Scripts. The former stores all the Scripts that will be used Server-side to process incoming Results and generate Alarms while the latter saves the Scripts that will be sent to the Probes, either for their initialization or for Scripts that will run on the Probe and measure the various aspects of this project. These tables have some columns that exist in both of them, with a Script being characterised with a title and the path to the file containing the Script, a field that states if the Script is considered Active or Inactive and a set of parameters to be used in the execution of the Scripts. By default, when uploading a Script the User is given the option to choose the Server Name, the Probe IP and a List of Targets to be added to the execution command but there is also a field to store additional parameters that might be relevant to that specific Script. A Script, similar to the Probe, has an owner and, as such, a field to store his ID. Finally, the table to store Monitoring Scripts contains an additional field to differentiate Scripts that are used in the initialization of the Probe from Scripts that should be running and performing tests. Associated with these Scripts, two more tables, Active Processing Scripts and Active Monitoring Scripts, store pairs of Probe ID and Script ID in order to give the information to the system where to run those same Scripts.

When the Monitoring Scripts are running at the Probes, they will collect measurements of some kind that will be stored in the Results table. This table is currently focused on RTT measurements and saves the timestamp for when the Result was taken as well the minimum, maximum, average

and the deviation obtained in the measurement. It also saves what Probe collected the Result, what was the Target of the measurement, what type of Result was collected and to what User this Result belongs to. It also has a field to flag if the Result is considered anomalous by the Result Management component. To store the various types of Results that can exist, an additional table stores the name for all of them and it is linked to the type field in the Result table.

In the previous paragraph, the Target was mentioned. This table stores the User that added the target and the IPv4 address for that Target. Finally, two tables store information regarding Alarms. The Alarm Target table stores information with regard to a specific target, stating the time of the Alarm, which Probe took the measurement and what was the Target Probed. The Alarm Probe table gives the information about what Probes are collecting anomalous measurements and, as such, stores only the timestamp and the Probe ID.

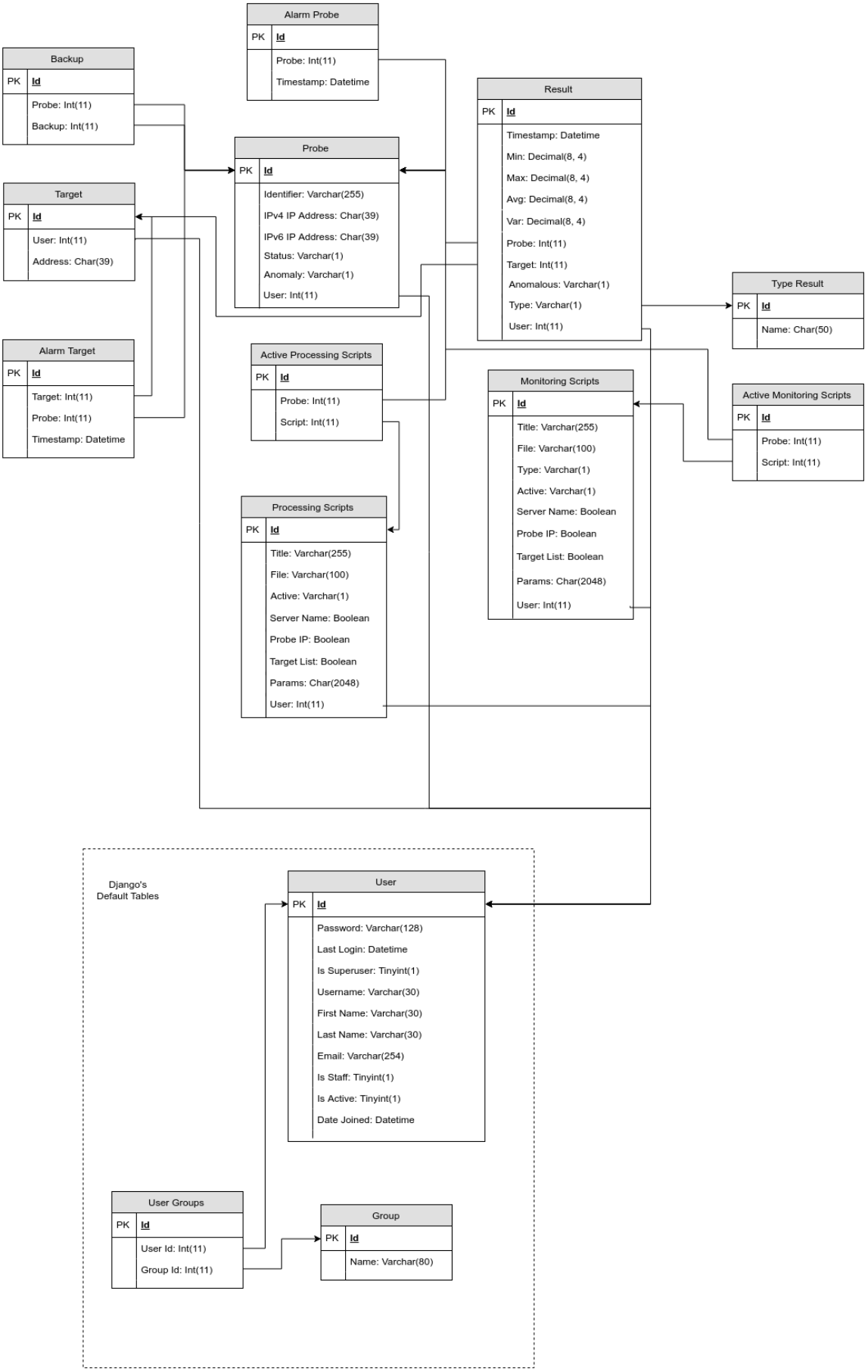


Figure 3.3: Datamodel used in the project

3.3 IMPLEMENTATION

In order to implement the components and the datamodel addressed in the previous sections, Python[47], in the version 2.7, was used as the programming language for the majority of the system with the addition of the Django Framework.

Django can be described as "a high-level Python Web framework that encourages rapid development and clean, pragmatic design"[48], especially focused on developers who want to set up their website quickly and with the basic functionalities already operating. When deploying a Django project for the first time, features as user registration and authentication, administration dashboard and database connections are already in place, leaving the developer to focus on other features that differentiate their systems. Inside of Django there are the concepts of project and apps, the former being the initial folder to be created by Django and containing the Settings file that will hold all the configurations necessary for the system, like the Database connector and credentials, apps installed to this project, middleware available, among others, and a Python file to map Uniform Resource Locators (URLs) to their corresponding apps, and the latter consisting of the web application that the developer is trying to implement. A project can contain more than one app and one app can be deployed to more than one project. To create this application, it is important to understand the concepts of Models, Views and Templates and how they can be linked together using the URLs files present in the project and in the app. This is actually close to the Model-View-Controller (MVC) design pattern, where "the model is the data, the view is the window on the screen, and the controller is the glue between the two."[49], however, the Django team explain themselves as a Model-Template-View (MTV) framework, where the View is responsible for what data is seen rather than how it is seen[50].

A Model is a set of instructions that define how data is stored, their names and types, and other functions that can be defined whenever necessary. After a model is defined, Django's Object-relational Mapper (ORM) translates them to the query language for the database in use. By default, Django is made to support only with Relational Databases but there are third-party extensions that can offer connections with some Non-Relational Databases, like MongoDB or Google App Engine. Simply put, Models allow developers to write their data model using Python code.

View functions, or Views for short, are Python functions that support the logical side of the application. Their input is a Request and they must output a Response, which may range from a 404 error to an image or, most commonly, the HTML content for a webpage[51]. It is up to the Views to collect the necessary data through the querysets made available from the Models and link it to a Template. After creating a View and making the connections to Models and Templates, the developer must declare the URL that will call that View, in order to make it accessible.

Lastly, the Templates create what users see on the website, populated with the information sent by the View in conjunction with the code present in them. These HTML Scripts can use, but is not mandatory, a templating engine so that they can dynamically render the information passed through the View. Django contains the Django Template Language (DTL) but it also supports Jinja2 [52] by default. As with the databases, other template engines can be supported using third-party extensions. Template files are able to support every HTML tag and/or the template engine tags, making them able to support JavaScript, CSS and other useful tools. This creates a file that can hold static information combined with loops, conditional clauses or objects from the database to dynamically assemble the Response that will be shown on the webpage.

3.3.1 DATABASE AND USER MODULE

The database is supported by MariaDB, an open-source system forked by the original developers of MySQL after its purchase by Oracle. MariaDB is intended to be compatible with MySQL binary files, making it easy to replace it, even if the original development was made under MySQL system. MariaDB is supported by default in Django and by using the Django's ORM and the built-in functions, the task of writing Structured Query Language (SQL) queries is removed from the developers. As such, all of the operations that interact with tables, columns or rows from the database are Python code.

When starting a new Django project, and depending on some libraries or middlewares that can be loaded into the Settings for the project, approximately 10 tables are created without the developer interference. These tables store information regarding Users, Groups, permissions, sessions, among other data and were used in this project as the User Module, removing the need to build one from zero.

Using these two components from the system, a set of three groups of users is defined in the system, with the following names and permissions:

Regular User - This group of users can only login, change information details for their account and list the existing Probes, Scripts and Alarms.

Server Admin - Adding to the Regular User permission, the Server Admin can also add, edit or delete Probes and Scripts.

Superuser - Created in Django itself, rather than in the system. These group of users can access the Django Admin Dashboard and manage Users and Groups of Users in the Authentication and Authorization module (default in Django) and manage Probes, Scripts, Results and Alarms in the Server module.

A Superuser must be created using the Linux terminal, in the folder where the file *manage.py* is present, with the following command:

```
$ python manage.py createsuperuser
```

And after inserting information regarding the username, email address and password for this superuser, a new account is created with enough privileges to access the `https://darteg.netconfs.com/admin` webpage. Also worth noting that, because Superusers are created on the command line rather than the website, after creation they must be integrated into the Server Admin group to have the same permissions regarding the view of the data.

3.3.2 RESULT AND MANAGEMENT COMMUNICATION

As presented before, the system contains two components that provide endpoints to the Probes interaction with the Server. Both the Result Communication as the Management Communication are built upon the Django REST Framework (DRF), a user-built Django app that describes itself as a "powerful and flexible toolkit for building Web APIs"[53] and it was used in this dissertation to support operations following a Representational State Transfer (REST) architecture. DRF offers a built-in browsable API by default, it can be linked to the database in use with the Django project to implement user authentication to the REST requests, serialisation of data based on JSON and several

layers of complexity in their classes to allow developers to use simple GET or POST requests to built their own logic in the API.

The Management Communication receives information from the Scripts that are tracking the resource utilisation at the Probes and can make changes to the status of the Probe in the database. This endpoint is located at `https://darteg.netconfs.com/rest/Probes/`, but should be addressed as a PUT request to `https://darteg.netconfs.com/rest/Probes/<Probe_id>` containing a JSON in the body of the request with the following format:

```
{
    "ipv4_ip_address": <string>,
    "status": <int>
}
```

The Probe ID in the URL in conjunction with the IP address in the body guarantee that is the correct Probe that will be modified in the database and the Status field states if the Probe is being activated (1) or deactivated (0). If a Backup Probe exists for the Probe being deactivated, it will be set as Active in order to ensure the monitorization in that area.

The other component, the Result Management, is responsible for accepting new Results, writing them in the database and alerting the Result Management and Alarms component that a new result needs to be processed. The Probes can access this component through the URL `https://darteg.netconfs.com/rest/results/` with a POST request that contains a JSON body with the subsequent structure:

```
{
    "op": <int>,
    "Probe": <string>,
    "target": <string>,
    "timestamp": <datetime>,
    "min": <decimal>,
    "max": <decimal>,
    "avg": <decimal>,
    "var": <decimal>
}
```

Where the "Op" signalises the operation made by the Probe, and at this stage of the project, only RTT measurements are supported which means that the "op" field should always be 1. Other options are discarded by the server. In the future, more options can be added to support other types of tests. The "probe" and "target" fields should both be IP addresses in the form of a string, the former serves the purpose of identification of the Probe who is making the measurements and the latter establishes what network was monitored in this measurement. The other five fields are straightforward, with the "timestamp" being the date and time of the readings and "max", "min", "avg" and "var" representing the maximum, minimum, average and the variance of the pings made by the Probe to the Target. After saving the Result to the database, the Result Communication alerts the Result Management,

through Python calls and passing the Result ID as part of the request.

3.3.3 PROBE MANAGEMENT

An important part of the system is the perception that the Server has of the various Probes. This is achieved by the Probe Management component, who is responsible for instantiating new Probes in the database, edit or delete them, and to upload Scripts, either for initializing the Probe or to run tests. The Probe Management also acts as the connection between the GUI, that needs User interaction, and the rest of the system, that works in an automated way.

After a User uploads a file containing a Python or Bash Script to the website it will trigger the Probe Management to start a task regarding that Script. A task, or background function, is a piece of Python code that is running "behind the scenes" while some other functions are also being executed, like, in the case of this project, the rendering of the website continues even if the task is running. These tasks are implemented with the help of Celery, a distributed task queue based on message passing to perform real-time operations but also able to support scheduling of operations[54]. A task queue is designed to distribute work (tasks) across multiple threads or machines, with the workers always listening to the queue until a new task arrives. To deliver this tasks, Celery uses a broker, which resolves the communication between the system using Celery and the workers, and supports a wide range of options. For this project, the broker used is Redis[55], a data structure store that can be implemented to work as a message queue, a cache or even a database, being classified as a key-value database (a Non-Relational model).

With this in mind, the Probe Management component has three tasks associated, two of them performing the transfer of Scripts between the Server and the Probes and one pinging the Probes to check for responsiveness. The latter is a really simple process that sends out Internet Control Message Protocol (ICMP) requests, more commonly known as pings, every 30 minutes and checks if the Probe is still responding to them, while the former divides into a function that sends one Script to various Probes, used after a new Script is uploaded to be used as a measurement Script, and another one that sends several Scripts to one Probe, used after a new Probe is instantiated in the system and needs to be initialized.

These connections are established using the *paramiko* library for Python, that provides helper functions to open an SSH channel to the Probe. After that, an SCP channel is opened over the transport provided by the SSH and the command is assembled with the parameters stored in the database before being executed over the same SSH channel.

Due to the real Probes used in this dissertation having different distributions of Linux-based Operating System (OS), with different package managers installed, the installation Scripts produced in the scope of the project must cover all possibilities. These Scripts are available in appendix B.

3.3.4 RESULT MANAGEMENT AND ALARMS

The last of the components that work autonomously is responsible for processing incoming Results and generate Alarms according to the Scripts present at the Server. This component is alerted by the Result Communication whenever a new Result is saved onto the database and it triggers a new task for processing the Result. Inside of this task, the database is consulted to see if there is any Script

uploaded by the User to process Results and, if the answer is negative, a default calculation is used, which is presented in subsection 3.3.5.

If a new User or developer wishes to implement a new method of processing Results, the database can be consulted by importing the correct Models and retrieving the necessary Results from the database. Django's ORM gives the option to retrieve one Result, all of the Results, or any number in between:

```
from .models import Result

results = Result.objects.all()                    # Retrieves all of the
    Results
results = Result.objects.filter(user=user_id)     # Retrieves the Results of
    the User with that ID
result = Result.objects.get(id=new_result_id)  # Retrieves the Result with
    that ID
```

Different ways to gather data from the database can be consulted using the Django's documentation[56].

On the other hand, after the calculations are made the Script must be able to save information back to the database in order to set alarms. As before, the import should be done to load the correct tables and then modify or create new entries in those tables:

```
from .models import Result, Alarm_Target, Alarm_Probe

result = Result.objects.get(id=result_id)    # Retrieves the Result with
    that ID
result.anomalous = '1'                       # Sets the processed result as
    anomalous
result.save()                                # Saves the changes to the
    database

from django.utils import timezone
new_alarm = Alarm_Target(target=target_id, Probe=Probe_id,
    timestamp=timezone.now())     # Creates a new Alarm regarding the
    target and Probe passed
new_alarm.save()                             # Saves the changes to the
    database

new_alarm = Alarm_Probe(Probe=Probe_id, timestamp=timezone.now())     #
    Creates a new Alarm regarding the target and Probe passed
new_alarm.save()                             # Saves the changes to the
    database
```

The *save* method should be called after all the modifications being made to ensure that they are persisted to the database.

3.3.5 DEFAULT RESULT PROCESSING

The default script for Result processing that was developed in the scope of this dissertation is based on the formulas present on the article written by P. Salvador and A. Nogueira[57]. The authors state that for each Probe p, who monitors n different networks during a time interval with t moments since its upstart, it is possible to define the sets H, R and D.

H contains all the time instants where the Probe p monitored network n in the interval $[t - \mathbf{H}, t[$, with **H** being a parameter that can be adjusted.

When a Probe p pings a network n, it will collect the times from that ping. These RTT will be stored in set R.

If a Result is found to contain an anomaly it will be part of the D and this deviation from the normal RTT is calculated by

$$R > \varepsilon \bar{r}$$

where $\bar{r}$ is

$$\bar{r} = \frac{1}{H} \sum_{t \in H \wedge t \notin D} R$$

This means that if the incoming result is higher than a constant ε times the average of the past H RTT, it is suffering a deviation. If K consecutive Results show a deviation, the Probe is considered as Anomalous and an Alarm is set on the database for Probe p, network n and timestamp t.

The epsilon used for this calculations, by default, is 1.2, k is 10 and H equals 480. This will force the Results to be processed using the last 480 Results for that Probe and that Target and since every ping is done with a 3 minute interval this means that the Script will use the Results in the last 24 hour period, if all of them are in a normal state. If a Result is found to be anomalous, it will be excluded from future calculations and force Results with more than 24 hours to appear in the set. The value for k represents the number of consecutive deviations necessary for an alarm to be set, which is 10, in this case. And the ε expresses the amount of error that is necessary for a Result to be considered anomalous in relation to his predecessors. The value used by default, 1.2, means that the result must be 20% higher than the average of the last RTTs to be considered anomalous. All of these values can be changed to provide different insights over the data already stored in the database.

3.3.6 WEBSITE AND GRAPHICAL USER INTERFACE

To ensure a way for the Users to interact with the whole system, a website with a GUI was developed using some of the tools provided by the Django framework. This website acts as an entrypoint for the data insertion and in the same time allows users to consult data about the tests that are being run and, if they have permission, to access the admin page in order to manage the Users and Groups present in the system.

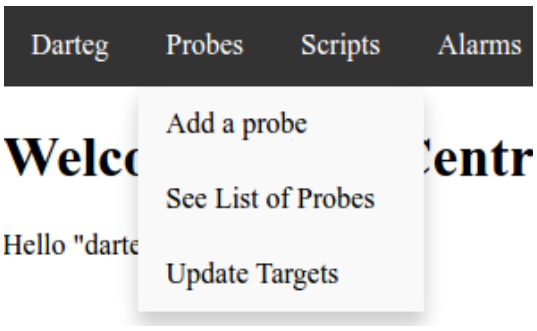
Every page in the website requires the user to be authenticated, redirecting the user to a login form, except the Index page, shown in figure 3.4, that welcomes the User and instructs him to login. The login form is a simple form containing a field for the username and another for the password. After the User confirms the inserted credentials, is up to Django and the User Module to concede permission to enter the website.

Figure 3.4: Index page of the website

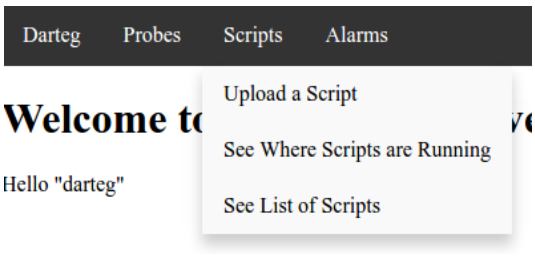
After the login process is complete, the User can start to access the various options offered by the site in the bar located at the top of the page. This bar, shown in detail in figure 3.5 divides itself into Probes, Scripts and Alarms, with the first two options expanding themselves to reveal the full set of links available. The CSS code responsible for this bar was done by Mário Pina.

Figure 3.5: Top bar of the website

Inside of the Probes group, a user with full privileges will see options to "Add a Probe", "See the List of Probes" and "Update Targets". In the Scripts group, it exists options for "Upload a Script", "See where Scripts are running" and "See a List of Scripts". Finally, the Alarms is a single link, not containing any more options. Figures 3.6a and 3.6b demonstrate these options.



(a) Options for the Probes option of the bar



(b) Options for the Scripts option of the bar

Figure 3.6: Options available in the bar of the website

Explaining every item in more detail, if the user selects the "Add a Probe" option a form will be shown with fields to create an identification tag (Identifier) for the Probe and to insert the IP address where the Probe is located. Additionally, a two option field must be selected in order to categorise the to-be added Probe as a Backup or an Active one. After the user confirms the submission, the Probe Management component is triggered in order to ensure the correct initialization of the Probes.

If the user chooses to see a "List of Probes", a list containing the ID number in the database, the identifier tag, the IP address for the Probe in question and its status is shown. If the user is a Server Admin, a hyperlink to Edit Probe and another to remove it are shown.

The "Edit Probe" uses the same form as the Add option, but with the fields already filled. If the edit is unintentional, the user can revert to the Probe list without any changes being made. The "Delete Probe" triggers a warning message to the user, where he must confirm that the deletion is intended. The actual process of deletion is handled by Django.

The final option, to "Update Targets", loads the current logged in user's target list from the database and displays it in a text box, with one target per line. The user can then edit them or even remove the whole list.

Advancing onto the Script options, if the user selects to "Upload a Script" a new blank form appears with seven fields to be filled. A text field to insert the title, a file prompt to select the Script from the user's file system, a two option field to categorise the Script as Active or Inactive (an Inactive Script will not be considered for execution) and a three options field about the type of execution for the Script, divided into "Probe init", "Probe run" and "Result analysis". This categorisation helps the server deciding the role of each uploaded Script. Assuming the Script is also Active, a "Probe init" Script is sent to a newly added Probe, using the "Add a Probe" option, a "Probe run" is sent to all active Probes after being uploaded and a "Result analysis" is kept in the server to be executed when a new result arrives from a Probe. All uploaded Scripts are written to the server's Script system, under a folder called "upload" located in the root directory of the project. If the Script added is of the type "Probe run" a task is immediately set after the upload. The last three fields are checkboxes that will state if the Script has any parameters to be added to the running command. Currently, this options only affect Python Scripts and are the Server Name, Probe's IP address and the List of user's targets.

Similar to the Probes functionalities, a "List of Scripts" option also exists, where a list with the title and the type of the Scripts is compiled to be shown to the user. With this list, options for seeing the contents of the Scripts and, if the user belongs to the "Server admin" group, editing the various Scripts, running it if the Script is of the type "Probe run" or remove the Script are also shown for each individual Script. The Scripts are shown grouped by their type follow the order "Probe init", "Probe run" and "Result analysis".

The "Show Script" feature calls a function created at the model level that parses the content of the Script, line by line, and shows the contents of the Script along with its title and path.

Again, similarly to the Probes, the "Edit Script" makes use of the form already used in the creation, but this time without the prompt to select a Script. This means that if the user mistakenly uploads an incorrect Script, they should do a new upload. Also, the "Delete Script" prompts a message to the user confirming that the deletion of the Script is the desired action, mimicking the behaviour of the "Delete Probe" function.

If the user already uploaded a Script but, for some reason, it is necessary to run it again on the active Probes, selecting the "See where Scripts are running" option will prompt the user to a new page where it is possible for the user to select a Script. After the selection is made, the system inquiries every Probe to see if the Script is actually running or not, and returns the result in two tables, for

"Running" or "Not Running". After that, the user has the option to move Probes from one table to another and to change the running parameters for that specific Script. After all the changes are made, a button in the bottom of the page allows the user to submit the changes. The majority of the functions and aspect of this page were a contribution from Mário Pina.

Lastly, the "Alarms" gathers the information about the alarms detected, showing which Probe triggered the alarm, for what target network the alarm was set and the timestamp when the alarm occurred. If no Alarms are available, a message is presented to the user stating this situation.

In case that the user is a Superuser, the Django Admin page is available. In this page, generated by default when a Django project is initiated, the user can access all records stored in the database, edit them and create new ones. This allows Superusers to manage the users registered in the system and their permissions, based on the Groups where they have a membership, and to supervise the data that is being stored in the database by the different actors in the system.

3.3.7 HTTPS SUPPORT

HTTPS, or HTTP over TLS[58], is a protocol that operates over Hypertext Transfer Protocol (HTTP) to guarantee authentication of the websites and the encryption of the data being transferred.

In the specific case of this project, HTTPS is implemented directly in the Apache's httpd configuration. Using the default port 443, httpd redirects every request received as plain HTTP to a *https* : // variant. The certificate used to support this protocol is a self-signed one, generated using the functionalities provided by OpenSSL and employing a 2048-bits RSA key. The full configuration can be found on Appendix A.

CHAPTER 4

Results

After the completion of the implementation phase of the project, the Server needed to be deployed in order to be available to the Users and a set of Probes installed in order to perform measurements. For this purpose, 20 different VPS were acquired and configured according to the requirements of the dissertation. On of these VPS was used to host the Server, with the remaining 19 acting as Probes. If two or more Probes are located in the same city, only one of them is configured as Active.

Table 4.1 details the capabilities of theses VPSs, such as, their memory, disk space, bandwidth and OS.

	City	Country	Cores	Memory (Mb)	Disk Space (Gb)	Bandwidth (Mbps)	Operating System
Server	Frankfurt	Germany	2	1024	20	500	Ubuntu 14.04
Probes	Frankfurt	Germany	1	1024	20	500	CentOS 6
	Chicago	U.S.A.	2	1024	20	500	CentOS 6
	Chicago	U.S.A.	1	1024	20	500	CentOS 6
	Los Angeles	U.S.A.	2	1024	20	500	CentOS 6
	Los Angeles	U.S.A.	1	1024	20	500	CentOS 6
	São Paulo	Brazil	2	1024	20	500	CentOS 6
	São Paulo	Brazil	1	1024	20	500	CentOS 6
	Johannesburg	South Africa	2	1024	20	500	CentOS 6
	Johannesburg	South Africa	1	1024	20	500	CentOS 6
		Chile	1	512	2	250	Debian 8
	Madrid	Spain	1	512	2	250	Debian 8
	Hong Kong	China	1	512	2	250	Debian 8
		Israel	1	512	2	250	Debian 8
		Iceland	1	512	2	250	Debian 8
	Milan	Italy	1	512	2	250	Ubuntu 14.04
	Amsterdam	Netherlands	1	512	2	250	Ubuntu 14.04
	Moscow	Russia	1	512	2	250	Debian 8
		Sweden	1	512	2	250	Debian 8
	London	United Kingdom	1	512	2	250	Debian 8

Table 4.1: Location, capabilities and OS of each VPS

As explained in subsection 3.3.3, these Probes are initialised with scripts uploaded by the Users, but for the scope of this dissertation, two scripts were already provided to them and they can be found on Appendix A.

In addition to the Probes, a set of Targets had to be selected to perform the RTT measurements for this project. These addresses, shown on table 4.2 were collected on one of many providers of servers and cloud services[59] and ensure a large enough set of Targets with enough disparity on the globe to provide meaningful measurements.

Location	IP Address
London	85.115.52.180
Frankfurt	85.115.56.180
Mumbai	116.50.59.180
Paris	85.115.60.180
Dusseldorf	85.115.58.180
Geneva	58.115.62.180
San Jose	208.87.233.180
Ashburn	208.87.234.180
Istanbul	85.115.32.180
Slough	85.115.54.180
Hong Kong	116.50.57.180
Sydney	116.50.58.180
Chicago	208.87.237.180
Dallas	208.87.239.180
Sao Paulo	177.39.96.180
Miami	208.87.238.180
Singapore	116.50.60.180
Johannesburg	196.216.238.180
Tokyo	116.50.61.180
Amsterdam	85.115.33.180

Table 4.2: Location and IP address of each target

4.1 RESULTS OBTAINED

Using the method described in the previous chapter, associated with the Probes and Targets presented before, the measurements of the RTTs ran since the 30th of June until the time of this writing. More than 600 thousand Results were collected and processed in order to discover anomalies and raise alarms.

Using the default values of $\varepsilon = 1,2$, $k = 10$ and $H = 480$, 3081 Results were flagged as anomalous and 120 Alarms were set, with the majority of them based on the Probe of Moscow to the Target in Tokyo and Probe of Chile to various targets.

If the threshold is lowered to $\varepsilon = 1,1$, the number of detections rises to 4055 anomalous Results with 419 Alarms.

To demonstrate the type of detection done in the scope of this dissertation, several graphics were prepared with the Results present in the database. Figures 4.1, 4.2 and 4.3 show measurements done from London, Hong Kong and Los Angeles to Chicago with the two thresholds mentioned before and figure 4.4 having the three RTT measurements in the same graphic. Similarly, figures 4.5, 4.6 and 4.7 give the same information from Moscow, Madrid and Chicago to Tokyo, with figure 4.8 grouping the three graphics in one.

Figure 4.1 clearly shows an anomaly happening during the 13th of July. However, the RTT before and after the anomaly drops from approximately 94 ms to 92 ms, which can lead to conclude that this is not an attack but rather a rerouting performed by the provider, with the spike in the RTT being, most likely, a convergence process in the network. Furthermore, analysing figure 4.4 shows that the other two Probes didn't observe any anomalies near that period.

On figure 4.2, only one anomaly is detected when using the default threshold, but when lowering this ceiling to $\varepsilon = 1,1$ anomalies are detected in, at least, two more occasions, with a third one happening near the 3rd of July but without sufficient data to set the threshold. When comparing with the other two probes, in figure 4.4, it is visible that the anomaly near the 11th of July is also captured by the Probe located in Los Angeles.

The last Probe analysed in this set is located in Los Angeles. Because of the proximity between the Probe and the Target, the RTT is very low, thus any deviation can provoke an anomaly. Nonetheless, near the 7th of July the values of the RTTs nearly quintupled, with two more occurrences during the first 15 days of July with the values rising four times more than the average. This can be explained by some type of load balacing, internal network congestion or other non-malicious factors, because on figure 4.4, only the anomaly on the 12th of July is detected by other Probe, Hong Kong.

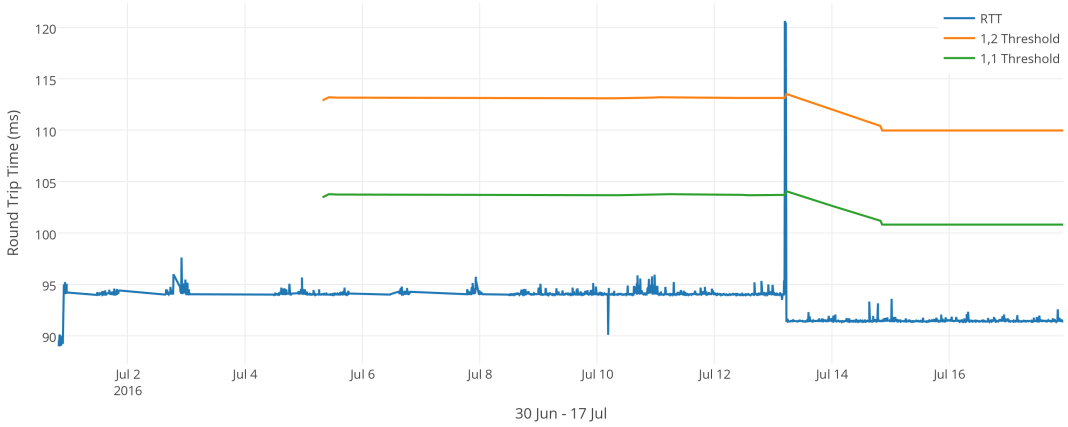


Figure 4.1: Measurements from Probe London to Target Chicago

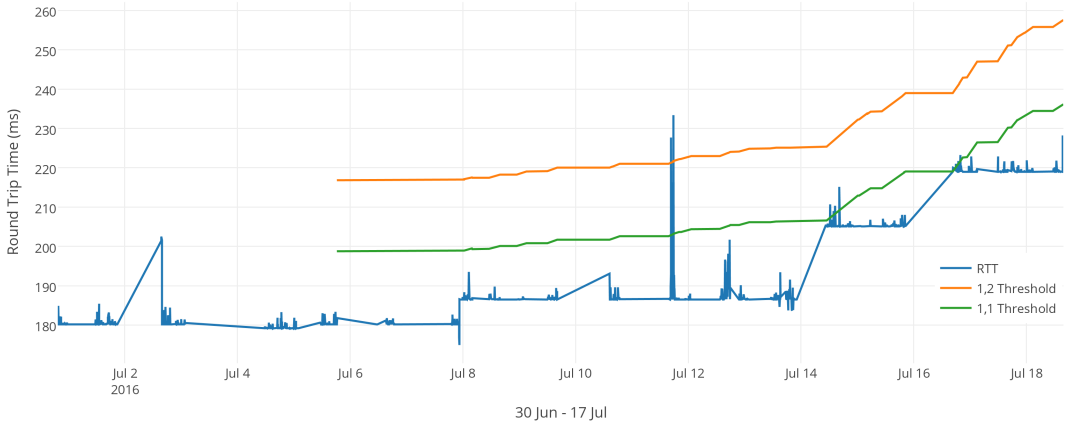


Figure 4.2: Measurements from Probe Hong Kong to Target Chicago

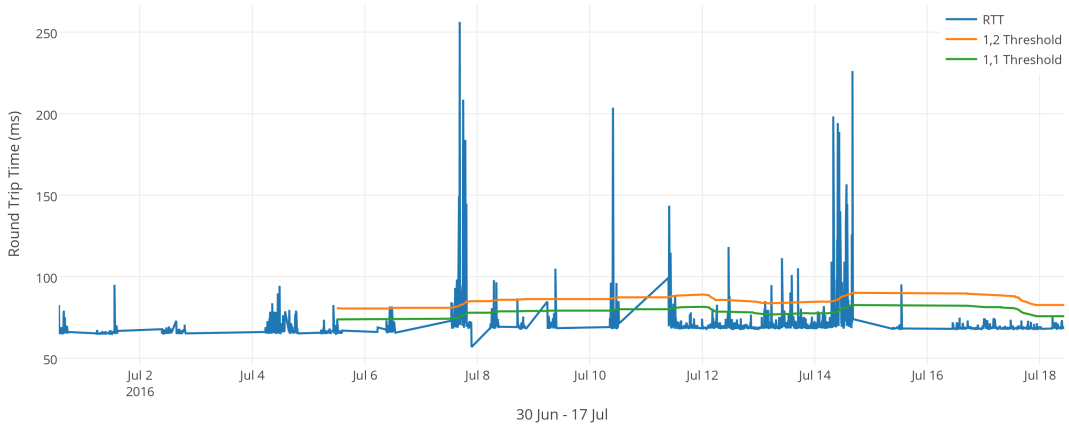


Figure 4.3: Measurements from Probe Los Angeles to Target Chicago

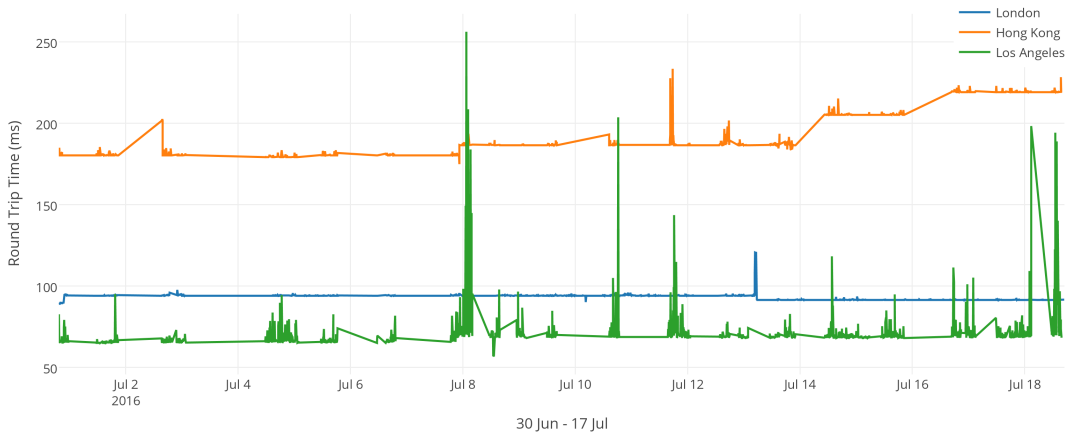


Figure 4.4: Measurements from Probes London, Hong Kong and Los Angeles to Target Chicago

Passing to the Target in Tokyo, figure 4.5 shows a curious situation near the 5th of July, where, with enough data, the system could detect an anomaly but further human observation can pinpoint this change in the RTT as a rerouting of traffic. This is further supported by figure 4.8 because no other Probe detects an anomaly on that date. Nonetheless, on the 8th of July, a spike in the values of the RTTs was not flagged by the system but figure 4.5 supports the theory of an anomaly happening because all three Probes show an increase in the values returned.

The Probe located in Madrid, represented in figure 4.6, shows a very homogeneous behaviour in the values, with the deviations from the average not exceeding 10 ms. The only situation worth noting was already approached in the previous paragraph.

Finally, in figure 4.7 shows the Probe located in Chicago having several isolated cases of anomalies but not enough consecutive ones to raise an Alarm. It also provides a drop in the RTT in the order of

10 ms during the 6th and 7th of July and then a permanent alteration during the night between the 13th and 14th of July, with no more changes being perceivable. This change is not reflected on any other Probe present on figure 4.8, which can lead to the conclusion of being an act conducted by the provider.

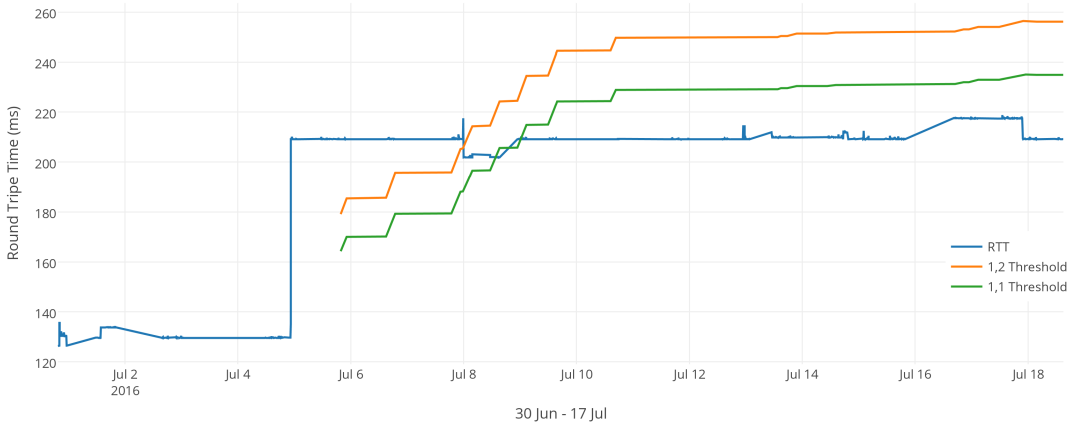


Figure 4.5: Measurements from Probe Moscow to Target Tokyo

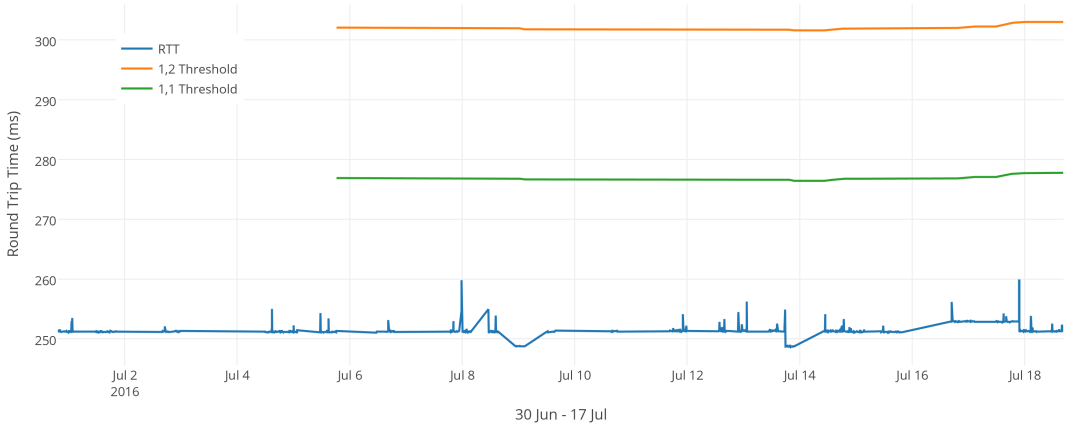


Figure 4.6: Measurements from Probe Madrid to Target Tokyo

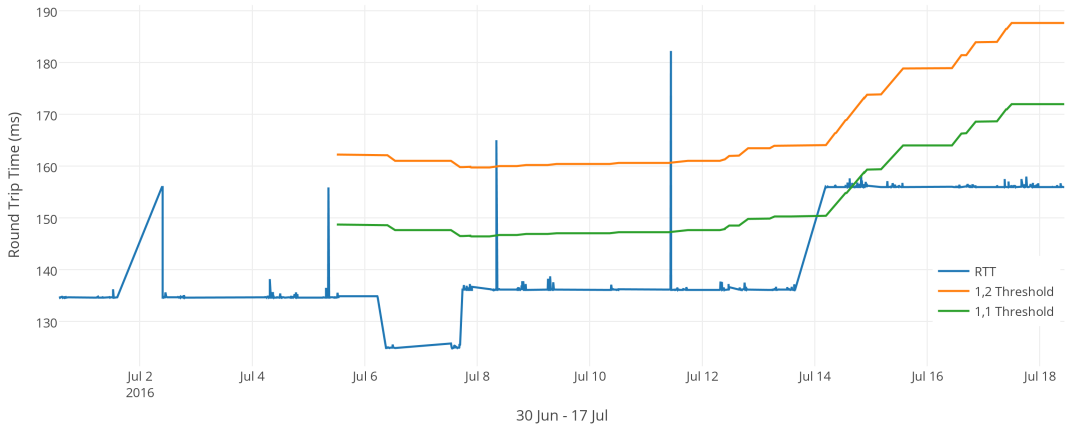


Figure 4.7: Measurements from Probe Chicago to Target Tokyo

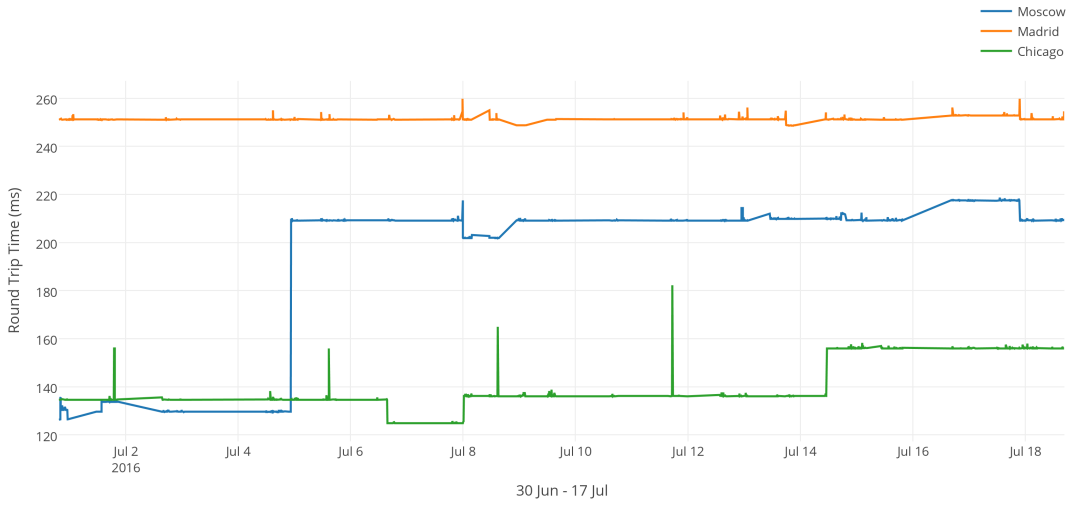


Figure 4.8: Measurements from Probes Moscow, Madrid and Chicago to Target Tokyo

CHAPTER 5

Conclusion

During the course of this dissertation, a system that is capable of monitoring a worldwide protocol using a low cost, easy to deploy solution was implemented.

Because of the flawed nature of BGP, it was necessary to build a system that would bring some information to users that could not access the BGP layer of their network, which represents all the domestic users and a large part of corporate clients.

Making use of 20 different VPSs spread through three continents, a Server and a set of Probes are constantly tracking the time it takes for the packet to flow on the Internet and trying to detect abnormal spikes in the travel time (RTT). Another dissertation was also involved in this system, in order to monitor the resource utilisation of the probes and to prevent the use of bad measurements in the Result analysis.

5.1 Future work

This work is subject to new features in order to expand the existing functionalities present in the Server. This could mean adding support to new measurements in the Probes, such as traceroutes, port mappings, Address Resolution Protocol (ARP) requests, among others, with the proper reception by the webservices, the usage of cryptographic keys to access the probes, which removes the need to use passwords in the server side, or simply the revamp of the look-and-feel of the website to further increase user interaction.

References

[1] *The cia triad: Confidentiality, integrity, availability*, http://panmore.com/the-cia-triad-confidentiality-integrity-availability, Accessed: 2016-07-01.

[2] *Darteg*, https://www.it.pt/Projects/Index/4257, Accessed: 2016-05-30.

[3] *Telecommunications and networking – av*, https://www.it.pt/Groups/Index/19, Accessed: 2016-05-30.

[4] C. Zheng, L. Ji, D. Pei, J. Wang, and P. Francis, "A light-weight distributed scheme for detecting ip prefix hijacks in real-time", *SIGCOMM Comput. Commun. Rev.*, vol. 37, no. 4, pp. 277–288, Aug. 2007, ISSN: 0146-4833. DOI: 10.1145/1282427.1282412. [Online]. Available: http://doi.acm.org/10.1145/1282427.1282412.

[5] Cisco Systems, Inc. (2008). Cisco bgp case studies, [Online]. Available: http://www.cisco.com/c/en/us/support/docs/ip/border-gateway-protocol-bgp/26634-bgp-toc.html (visited on 02/05/2016).

[6] Y. Rekhter, T. Li, and S. Hares, *A border gateway protocol 4 (bgp-4)*, RFC 4271 (Draft Standard), Updated by RFCs 6286, 6608, 6793, Internet Engineering Task Force, Jan. 2006. [Online]. Available: http://www.ietf.org/rfc/rfc4271.txt.

[7] J. Hawkinson and T. Bates, *Guidelines for creation, selection, and registration of an autonomous system (as)*, RFC 1930 (Best Current Practice), Updated by RFCs 6996, 7300, Internet Engineering Task Force, Mar. 1996. [Online]. Available: http://www.ietf.org/rfc/rfc1930.txt.

[8] Q. Vohra and E. Chen, *Bgp support for four-octet as number space*, RFC 4893 (Proposed Standard), Obsoleted by RFC 6793, Internet Engineering Task Force, May 2007. [Online]. Available: http://www.ietf.org/rfc/rfc4893.txt.

[9] A. Pilosov and T. Kapela. (2008). Stealing the internet: An internet-scale man in the middle attack, [Online]. Available: https://www.defcon.org/images/defcon-16/dc16-presentations/defcon-16-pilosov-kapela.pdf (visited on 02/05/2016).

[10] V. Giotsas and S. Zhou, "Valley-free violation in internet routing - analysis based on bgp community data.", in *ICC*, IEEE, 2012, pp. 1193–1197, ISBN: 978-1-4577-2052-9. [Online]. Available: http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=6363987&tag=1.

[11] E. Zmijewski. (2009). To catch a thief, [Online]. Available: http://research.dyn.com/2009/02/stealing-the-internet-back-1/ (visited on 03/12/2016).

[12] R. Chandra, P. Traina, and T. Li, *Bgp communities attribute*, RFC 1997 (Proposed Standard), Internet Engineering Task Force, Aug. 1996. [Online]. Available: http://www.ietf.org/rfc/rfc1997.txt.

[13] Cisco Systems, Inc. (2008). Cisco bgp case studies, [Online]. Available: http://www.cisco.com/c/en/us/support/docs/ip/border-gateway-protocol-bgp/26634-bgp-toc.html#bgpconfed (visited on 02/10/2016).

[14] C. Hepner, E. Zmijewski, and Renesys Corporation. (2009). Defending against bgp man-in-the-middle attacks, [Online]. Available: http://research.dyn.com/wp-content/uploads/2013/05/blackhat-09.pdf (visited on 02/11/2016).

[15] D. Madory and Dyn Research. (2015). The vast world of fraudulent routing, [Online]. Available: http://research.dyn.com/2015/01/vast-world-of-fraudulent-routing/ (visited on 02/11/2016).

[16] ——, (2014). Sprint, windstream: Latest isps to hijack foreign networks, [Online]. Available: http://research.dyn.com/2014/09/latest-isps-to-hijack/ (visited on 02/11/2016).

[17] ——, (2013). Intrigue surrounds smw4 cut, [Online]. Available: http://research.dyn.com/2013/03/intrigue-surrounds-smw4-cut/ (visited on 02/11/2016).

[18] J. Cowie and Dyn Research. (2011). Japan quake, [Online]. Available: http://research.dyn.com/2011/03/japan-quake/ (visited on 02/11/2016).

[19] M. Brown and Dyn Research. (2008). Pakistan hijacks youtube, [Online]. Available: http://research.dyn.com/2008/02/pakistan-hijacks-youtube-1/ (visited on 02/11/2016).

[20] S. T. Zargar, J. Joshi, and D. Tipper, "A survey of defense mechanisms against distributed denial of service (ddos) flooding attacks.", *IEEE Communications Surveys and Tutorials*, vol. 15, no. 4, pp. 2046–2069, 2013.

[21] Anonymous, "The collateral damage of internet censorship by dns injection", *SIGCOMM Comput. Commun. Rev.*, vol. 42, no. 3, pp. 21–27, Jun. 2012, ISSN: 0146-4833. DOI: 10.1145/2317307.2317311. [Online]. Available: http://doi.acm.org/10.1145/2317307.2317311.

[22] S. Goldberg, "Why is it taking so long to secure internet routing?", *Queue*, vol. 12, no. 8, 20:20–20:33, Aug. 2014, ISSN: 1542-7730. DOI: 10.1145/2668152.2668966. [Online]. Available: http://doi.acm.org/10.1145/2668152.2668966.

[23] E. Zmijewski. (2014). The end of undetected bgp route hijacking (detecting malicious behavior using global data analytics), [Online]. Available: http://research.dyn.com/wp-content/uploads/2014/05/Linx851.pdf (visited on 02/15/2016).

[24] Cisco Systems, Inc. (2015). Cisco bgp best path selection algorithm, [Online]. Available: http://www.cisco.com/c/en/us/support/docs/ip/border-gateway-protocol-bgp/13753-25.html#bestpath (visited on 02/15/2016).

[25] Juniper Networks, Inc. (2013). Understanding bgp path selection, [Online]. Available: http://www.juniper.net/techpubs/en_US/junos13.1/topics/reference/general/routing-ptotocols-address-representation.html (visited on 02/15/2016).

[26] J. Cowie. (2015). The good, bad, and ugly of internet routing, [Online]. Available: http://research.dyn.com/wp-content/uploads/2014/07/Cowie-Stanford-EE380-sm.pdf (visited on 02/17/2016).

[27] C. E. Landwehr, A. R. Bull, J. P. McDermott, and W. S. Choi, "A taxonomy of computer program security flaws", *ACM Comput. Surv.*, vol. 26, no. 3, pp. 211–254, Sep. 1994, ISSN: 0360-0300. DOI: 10.1145/185403.185412. [Online]. Available: http://doi.acm.org/10.1145/185403.185412.

[28] R. A. Grimes. (2012). The 5 cyber attacks you're most likely to face, [Online]. Available: http://www.infoworld.com/article/2616316/security/the-5-cyber-attacks-you-re-most-likely-to-face.html (visited on 02/18/2016).

[29] J. Cowie and Dyn Research. (2013). The new threat: Targeted internet traffic misdirection, [Online]. Available: http://research.dyn.com/2013/11/mitm-internet-hijacking/ (visited on 02/29/2016).

[30] D. Madory and Dyn Research. (2015). Uk traffic diverted through ukraine, [Online]. Available: http://research.dyn.com/2015/03/uk-traffic-diverted-ukraine/ (visited on 03/03/2016).

[31] S. Kent, C. Lynn, L. Sanchez, M. Steenstrup, M. Casagni, and K. Seo, *Bgp countermeasures (secure-bgp)*, Internet Engineering Task Force, 1998. [Online]. Available: http://www.ir.bbn.com/sbgp/IETF42.ppt.

[32] Comodo Group, Inc. (). What is pki?, [Online]. Available: https://www.comodo.com/resources/small-business/digital-certificates1.php (visited on 02/22/2016).

[33] Internetwork Research group. (). Secure bgp project (s-bgp), [Online]. Available: http://www.ir.bbn.com/sbgp/ (visited on 02/22/2016).

[34] R. Zouaghi and S. Wolthusen. (). Interdomain routing security (bgp-4), [Online]. Available: http://cdn.ttgtmedia.com/searchSecurityUK/downloads/RHUL_Z_final.pdf (visited on 02/22/2016).

[35] A. Retana. (2003). Secure origin bgp (sobgp), [Online]. Available: https://www.nanog.org/meetings/nanog28/presentations/alvaro.pdf (visited on 02/22/2016).

[36] G. Huston and R. Bush. (2011). Securing bgp with bgpsec, [Online]. Available: http://www.potaroo.net/ispcol/2011-07/bgpsec.pdf (visited on 02/23/2016).

[37] M. Lepinski, *Bgpsec protocol specification*, Internet Engineering Task Force, Dec. 2015. [Online]. Available: https://tools.ietf.org/id/draft-ietf-sidr-bgpsec-protocol-14.txt.

[38] R. Zouaghi. (2009). Interdomain routing security (bgp-4) a comparison between s-bgp and sobgp, [Online]. Available: http://www.ma.rhul.ac.uk/static/techrep/2009/RHUL-MA-2009-01.pdf (visited on 02/23/2016).

[39] Z. Zhang, Y. Zhang, Y. C. Hu, Z. M. Mao, and R. Bush, "Ispy: Detecting ip prefix hijacking on my own", *SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 4, pp. 327–338, Aug. 2008, ISSN: 0146-4833. DOI: 10.1145/1402946.1402996. [Online]. Available: http://doi.acm.org/10.1145/1402946.1402996.

[40] M. Lad, D. Massey, D. Pei, Y. Wu, B. Zhang, and L. Zhang, "Phas: A prefix hijack alert system", in *Proceedings of the 15th Conference on USENIX Security Symposium - Volume 15*, ser. USENIX-SS'06, Vancouver, B.C., Canada: USENIX Association, 2006. [Online]. Available: http://dl.acm.org/citation.cfm?id=1267336.1267347.

[41] Z. Zhang, Y. Zhang, Y. C. Hu, and Z. M. Mao, "Practical defenses against bgp prefix hijacking", in *Proceedings of the 2007 ACM CoNEXT Conference*, ser. CoNEXT '07, New York, New York: ACM, 2007, 3:1–3:12, ISBN: 978-1-59593-770-4. DOI: 10.1145/1364654.1364658. [Online]. Available: http://doi.acm.org/10.1145/1364654.1364658.

[42] *Internet alert registry*, http://www.cs.unm.edu/~karlinjf/IAR/, Accessed: 2016-02-29.

[43] *Routing information service (ris)*, https://www.ripe.net/analyse/internet-measurements/routing-information-service-ris, Accessed: 2016-02-29.

[44] *Bgpmon*, http://www.bgpmon.net/, Accessed: 2016-02-29.

[45] *Internet intelligence*, http://dyn.com/internet-intelligence/, Accessed: 2016-02-29.

[46] *Watchmy.net*, http://www.watchmy.net/, Accessed: 2016-02-29.

[47] *Welcome to python.org*, https://www.python.org/, Accessed: 2016-07-15.

[48] *Django*, https://www.djangoproject.com/, Accessed: 2016-05-24.

[49] *Model view controller*, http://c2.com/cgi/wiki?ModelViewController, Accessed: 2016-05-26.

[50] *Model template view*, https://docs.djangoproject.com/en/1.9/faq/general/#django-appears-to-be-a-mvc-framework-but-you-call-the-controller-the-view-and-the-view-the-template-how-come-you-don-t-use-the-standard-names, Accessed: 2016-05-26.

[51] *Django views*, https://docs.djangoproject.com/en/1.9/topics/http/views/, Accessed: 2016-05-26.

[52] *Jinja2*, http://jinja.pocoo.org/docs/dev/, Accessed: 2016-05-26.

[53] *Django rest framework*, http://www.django-rest-framework.org/, Accessed: 2016-05-24.

[54] *Celery*, http://www.celeryproject.org/, Accessed: 2016-05-26.

[55] *Redis*, http://redis.io/, Accessed: 2016-05-26.

[56] *Django queryset api reference*, https://docs.djangoproject.com/en/1.9/ref/models/querysets/, Accessed: 2016-05-30.

[57] P. Salvador and A. Nogueira, "Customer-side detection of internet-scale traffic redirection", in *Telecommunications Network Strategy and Planning Symposium (Networks), 2014 16th International*, Funchal, Madeira, 2014. DOI: 10.1109/NETWKS.2014.6958532. [Online]. Available: http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=6958532.

[58] E. Rescorla, *Http over tls*, RFC 2818 (Informational), Updated by RFCs 5785, 7230, Internet Engineering Task Force, May 2000. [Online]. Available: http://www.ietf.org/rfc/rfc2818.txt.

[59] *Cloud service data center (cluster) ip addresses and port numbers*, http://www.websense.com/support/article/kbarticle/Cloud-service-data-center-IP-addresses-port-numbers, Accessed: 2016-07-10.

Appendix A: Server Installation Manual

In order to install the Server in another VPS, different packages and libraries must installed. In the aptitude package manager (or the equivalent for the OS used):

```
$ apt-get install python-pip
$ apt-get install apache2
$ apt-get install libapache2-mod-wsgi
$ apt-get install libffi-dev
$ apt-get install python-dev
$ apt-get install mariadb-server
$ apt-get install libmariadbclient-dev
$ apt-get install libssl-dev
```

And in the Python Package Manager, also known as pip, the virtual environment module should be installed and configured:

```
$ pip install virtualenv
$ virtualenv dartegenv
$ source dartegenv/bin/activate
$ pip install django
$ pip install djangorestframework
$ pip install markdown
$ pip install django-filter
$ pip install celery
$ pip install redis
$ pip install mysqlclient
$ pip install paramiko
$ pip install scp
$ pip install pycrypto
```

To install MariaDB, there is a need to configure the database with:

```
$ mysql_secure_installation
```

```
$ mysql -u root -p
```

And then, inside the MariaDB shell:

```
CREATE DATABASE <name> CHARACTER SET UTF8;
CREATE USER <user>@localhost IDENTIFIED BY <password>;
GRANT ALL PRIVILEGES ON <name>.* TO <user>@localhost;
FLUSH PRIVILEGES;
```

Finally, in the folder that stores the Django code, the following commands should be applied:

```
$ python manage.py makemigrations
$ python manage.py migrate
$ python manage.py createsuperuser
```

However, if the Server is to be served by httpd, to mimic this project, a conf.d file should be prepared to handle *http* and *https* requests. For this project, the following configuration was used:

```
<VirtualHost *:80>
        ServerAdmin webmaster@localhost
        DocumentRoot /var/www/html/darteg

        ErrorLog ${APACHE_LOG_DIR}/error.log
        CustomLog ${APACHE_LOG_DIR}/access.log combined

        RewriteEngine On
        RewriteCond %{SERVER_PORT} !^443$
        RewriteRule ^(.*)$ https://%{HTTP_HOST}$1 [R=301,L]

        ServerName darteg.netconfs.com
</VirtualHost>

<VirtualHost *:443>
        ServerName darteg.netconfs.com

        Alias /static /var/www/html/darteg/static
        <Directory /var/www/html/darteg/static>
                Require all granted
        </Directory>

        <Directory /var/www/html/darteg/darteg/>
                <Files wsgi.py>
                        Require all granted
                </Files>
        </Directory>

        WSGIDaemonProcess darteg
            python-path=/var/www/html/darteg:/var/www/html/darteg/dartegenv/lib/python2.7/site-
```

```
        WSGIProcessGroup darteg
        WSGIScriptAlias / /var/www/html/darteg/darteg/wsgi.py
            process-group=darteg
        WSGIScriptReloading On

        SSLEngine on
        SSLCertificateFile /var/www/html/darteg/cert.pem
        SSLCertificateKeyFile /var/www/html/darteg/private.pem
        SSLCACertificateFile /var/www/html/darteg/cert.pem
</VirtualHost>

To generate a self-signed certificate to use as the SSL certificate, the
    following command must be executed:

\begin{lstlisting}[language=bash]
  $ openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout
      /etc/ssl/private/apache-selfsigned.key -out
      /etc/ssl/certs/apache-selsigned.crt
```

This configuration file will take all *http* requests and redirect them to the *https* port. In that port, the file has the configurations needed to fetch the correct files for the project, as well as the certificate and private key used in it. After this file is placed in the sites-avaialable folder of apache, the following commands finish the process:

```
  $ chown :www-data ~/darteg
  $ a2ensite <configuration_file>.conf
  $ python manage.py collectstatic
  $ service apache2 restart
```

The Server is now ready to be used.

Appendix B: Probe Initialisation Scripts

To ensure the correct functioning of the Probes when the measurement scripts arrive, and because of the different OSs present in this project, two minor scripts with packages to be installed were created.

For Debian-based OSs:

```
$ apt-get install -y python2.7
$ apt-get install -y python-pip
$ pip install -y requests
```

And for Probes with CentOS:

```
$ yum install -y epel-release
$ yum install -y python-pip
$ pip install requests
```